НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК 004.9

«До захисту допущено»
В.о. завідувача кафедри
_____ О.І. Ролік
«___»_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

на тему: «Дослідження впливу застосування спайкових нейронних мереж при навчанні класичних»

Виконав:
студент ІІ курсу, групи ІА-61м
Колісніченко Вадим Юрійович _____

Керівник:
Доцент, к.т.н, доцент
Дорогий Я.Ю. _____

Рецензент:
Доц. каф. кібербезпеки та ЗІСТ, к.т.н., доцент
Цуркан В.В. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ С.Ф. Теленик

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Колісніченку Вадиму Юрійовичу

1. Тема дисертації «Дослідження впливу застосування спайкових нейронних мереж при навчанні класичних», науковий керівник дисертації Дорогий Ярослав Юрійович, доцент, к.т.н., доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження – використання спайкових нейронних мереж в роботі штучних нейронних мережі.

4. Предмет дослідження – застосування спайкових нейронних мереж на етапі попереднього тренування при навчанні класичних нейронних мереж.

5. Перелік завдань, які потрібно розробити: аналіз компонентів штучних та спайкових нейронних мереж; аналіз методів попереднього тренування; розробка методу попереднього тренування за допомогою спайкових нейронних мереж; визначення компонентів розробленого методу попереднього тренування, які дають найвищу точність штучної нейронної мережі; написання пояснювальної записки.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: структура побудованої нейронної мережі; схема з'єднання між нейронами двох шарів нейронної мережі; ваги нейронної мережі.

7. Перелік публікацій: 1) «Аналіз моделей спайкових нейронів» – Міжнародна науково-технічна конференція «Сучасні інформаційно-телекомунікаційні технології» – 2015 2) «Дослідження методів тренування спайкових нейронних мереж» – II

Міжнародна науково-технічна конференція «Актуальні проблеми розвитку науки і техніки» – 2015 3) «Designing Spiking Neural Networks» – Proceedings of the XIIIth International Conference TCSET'2016.

8. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз компонентів штучних та спайкових нейронних мереж	15.03.18-28.03.18	
2	Аналіз методів попереднього тренування	29.03.18-08.04.18	
3	Розробка методу попереднього тренування за допомогою спайкових нейронних мереж	09.04.18-24.04.18	
4	Визначення компонентів розробленого методу попереднього тренування, які дають найвищу точність штучної нейронної мережі	25.04.18-01.05.18	
5	Написання пояснювальної записки	02.05.18-14.05.18	

Студент В.Ю. Колісніченко

Науковий керівник дисертації Я.Ю. Дорогий

РЕФЕРАТ

Магістерська дисертація освітньо-кваліфікаційного рівня “магістр” на тему “Дослідження впливу застосування спайкових нейронних мереж при навчанні класичних”: 125с., 39 рис., 26 табл., 3 додатки, 30 джерел.

Об'єкт дослідження – використання спайкових нейронних мереж в роботі штучних нейронних мереж.

Мета роботи – дослідити вплив застосування спайкових нейронних мереж на етапі попереднього тренування на точність навчання класичних штучних нейронних мереж.

Штучні нейронні мережі зазвичай вимагають розмічених наборів даних для вирішення найбільш цікавих проблем. Збір позначених даних зазвичай є складною проблемою, оскільки потребує ручної анотації. А велика кількість нерозмічених даних, навпаки, може бути легко зібрана автоматично.

Ідея використовувати нерозмічені дані для покращення показників штучних нейронних мереж при навчанні на розмічених даних не є новою. Нерозмічений набір даних, як правило, застосовується перед навчанням з учителем, на етапі попереднього тренування. Подібний принцип також застосовується до завдань напівавтоматичного навчання, коли кількість розмічених прикладів мала.

У даній роботі досліджується вплив застосування нового розробленого підходу до задач напівавтоматичного навчання традиційних штучних нейронних мереж, який базується на використанні спайкових нейронних мереж на етапі попереднього тренування.

Прогнозні припущення про розвиток дослідження – розробка ефективнішого алгоритму перенесення моделі спайкової нейронної мережі та застосування запропонованого методу попереднього тренування для більш складних архітектур штучних нейронних мереж.

СПАЙКОВІ НЕЙРОННІ МЕРЖІ, НАПІВАВТОМАТИЧНЕ НАВЧАННЯ, НАВЧАННЯ БЕЗ УЧИТЕЛЯ, ПОПЕРЕДНЄ ТРЕНУВАННЯ.

ABSTRACT

Master's thesis “Machine learning frameworks for pattern recognition” consists of 125 pages, 39 figures, 26 tables, 3 appendices, 30 sources.

The object of the study is the use of spiking neural networks in the work of artificial neural networks.

The purpose of the work is to analyze the effect of the use of spiking neural networks in the pre-training phase on the accuracy of classical artificial neural networks.

Artificial neural networks usually require labeled data sets for solving the most interesting problems. Collecting labeled data is usually a complex issue, because it requires manual annotation. A large number of unlabeled data, in contrast, can be easily collected automatically.

The idea of using unlabeled data to improve the performance of artificial neural networks when learning on labeled data is not new. An unlabeled data set is usually applied before supervised learning, at the stage of the pre-training. A similar principle is also applicable to the tasks of semi-supervised learning, when the number of labeled examples is small.

In this paper, the effect of the application of the new developed approach to the tasks of semi-supervised learning of traditional artificial neural networks, based on the use of spiking neural networks at the pre-training phase, is being studied.

Foreseeable assumptions about the development of the study – development of a more efficient algorithm for transferring the model of the spiking neural network and the application of the proposed method of pre-training for more complex architectures of artificial neural networks.

SPIKING NEURAL NETWORKS, SEMI-SUPERVISED LEARNING, UNSUPERVISED LEARNING, PRE-TRAINING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОЛИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БП - безумовний подразник

УС - умовний стимул

ABS - Artola, Bröcher, Singer

AE - autoencoder

ANN - artificial neural network

BM - Boltzman machine

CNN - convolutional neural network

ESN - echo state network

FFNN - feed forward neural network

HH - Hodgkin-Huxley

HN - Hopfield network

IF - integrate-and-fire

LIF - leaky-integrate-and-fire

LSM - liquid-state machine

RBM - restricted Boltzmann machine

ReLU - rectified linear unit

ReSuMe - remote supervised method

RNN – recurrent neural network

SNN – spiking neural network

SRM - spike response model

STDP - spike-timing-dependent plasticity

ВСТУП

Актуальність. Актуальність теми полягає у тому, що штучні нейронні мережі зазвичай вимагають маркованих (розмічених) наборів даних для вирішення найбільш цікавих проблем, а використовуючи спайкові нейронні мережі можна навчатись на нерозмічених даних. Збір розмічених даних зазвичай є складною проблемою, оскільки для цього потрібна ручна анотація. А велика кількість нерозмічених даних, навпаки, може бути легко зібрана автоматично. Було б непогано використовувати нерозмічені дані, щоб підвищити точність штучної нейронної мережі.

Ідея використовувати нерозмічені дані для покращення показників штучних нейронних мереж при навчанні на розмічених даних не є новою і була реалізована багато разів. Нерозмічений набір даних, як правило, застосовується перед контрольованим навчанням (з учителем) на етапі попереднього тренування. Подібна стратегія також може застосовуватися до завдань напівавтоматичного навчання, коли кількість розмічених прикладів відносно невелика. Існуючі методи не є досконалими і потребують покращення або створення принципово нових методів.

Зв’язок роботи з науковими програмами, планами, темами. Тематика роботи включена в науково-технічні плани кафедри автоматики та управління в технічних системах Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Метою дослідження є аналіз впливу застосування спайкових нейронних мереж на етапі попереднього тренування на точність навчання класичних штучних нейронних мереж.

Задачами дослідження є аналіз компонентів штучних та спайкових нейронних мереж, аналіз методів попереднього тренування, розробка методу попереднього тренування за допомогою спайкових нейронних мереж, визначення компонентів розробленого методу попереднього тренування, які дають найвищу точність штучної нейронної мережі.

Об’єктом дослідження є використання спайкових нейронних мереж в роботі штучних нейронних мережі.

Предметом дослідження є застосування спайкових нейронних мереж на етапі попереднього тренування при навчанні класичних нейронних мереж.

Методами дослідження, що використовуються для досягнення мети є методи обробки зображень, методи класифікації та розпізнавання, методи ініціалізації ваг нейронної мережі, методи регуляризації та оптимізації, методи та алгоритми навчання нейронних мереж.

Новизна магістерського дослідження полягає у розробці нового методу попереднього тренування на основі спайкових нейронних мереж та аналізі впливу запропонованого методу на точність штучної нейронної мережі.

Практичне значення полягає у тому, що якщо розширити запропонований підхід на більш складні архітектури нейронних мереж, то це дозволить вирішувати актуальні задачі з малою кількістю розмічених даних.

Апробація результатів. Результати дослідження апробовані і опубліковані у вигляді тез «Аналіз моделей спайкових нейронів» на Міжнародній науково-технічній конференції «Сучасні інформаційно-телекомунікаційні технології» (додаток А), «Дослідження методів тренування спайкових нейронних мереж» на ІІ Міжнародній науково-технічній конференція «Актуальні проблеми розвитку науки і техніки» (додаток Б) та у вигляді статті «Designing Spiking Neural Networks» в «Proceedings of the XIIIth International Conference TCSET’2016» (додаток В).

1 МАШИННЕ НАВЧАННЯ ТА НЕЙРОННІ МЕРЕЖІ

1.1 Типи машинного навчання

Труднощі, з якими стикаються системи, що спираються на жорстко-кодовані знання, дозволяють стверджувати, що системи штучного інтелекту потребують можливості навчатися шляхом виділення шаблонів з вихідних даних. Ця можливість називається машинним навчанням.

Алгоритми машинного навчання організовані в групи, на основі бажаного результату алгоритму [1]. Загальні типи алгоритмів включають:

- Навчання з учителем (контрольоване навчання, supervised learning) - де алгоритм генерує функцію, яка відображає входи до потрібних виходів (результатів). Одним стандартним формулюванням контрольованого навчання є завдання класифікації: навчаючий повинен вивчати (для наближення поведінки) функцію, яка відображає вектор в один з декількох класів, дивлячись на декілька прикладів вводу-виводу функції;

- Навчання без вчителя (безконтрольне навчання, unsupervised learning) - моделює набір входів: відмічені приклади не доступні;

- Навчання з підкріпленням (reinforcement learning) - алгоритм вивчає правила, як діяти з огляду на середовище. Кожна дія впливає на навколишнє середовище, і навколишнє середовище забезпечує зворотний зв’язок, який спрямовує алгоритм навчання.

Ці типи навчання вважаються основними, але можна виділити ще один - напівавтоматичне навчання (semi-supervised learning). Воно поєднує як позначені, так і непозначені дані для створення відповідної функції або класифікатора. Цей тип можна назвати середнім між навчанням з учителем та навчанням без учителя.

Машинне навчання полягає у розробці алгоритмів, які дозволяють комп’ютеру вчитися. Навчання не обов’язково включає в себе свідомість, але навчання - це питання пошуку статистичних закономірностей або інших моделей даних. Таким чином, багато алгоритмів машинного навчання трохи нагадують, як людина може

підійти до навчального завдання. Однак алгоритми навчання можуть дати розуміння відносної складності навчання в різних середовищах.

1.1.1 Навчання з учителем

За допомогою керованого навчання подається вихід функції у систему. Це означає, що в навчанні з учителем машина вже знає вихід функції, перш ніж вона почне працювати на ньому або навчатися йому [2]. Основним прикладом цього поняття буде студент, який вивчає курс інструктора. Студент знає, що він/вона навчається за програмою курсу.

З вихідним значенням функції все, що потрібно зробити системі, полягає в тому, щоб виробити кроки чи процес, необхідний для досягнення такого ж відображення від входу до виводу. Алгоритм навчається через набір навчальних даних, який керує машиною (рис. 1.1). Якщо процес ішов у бік, а алгоритми виходять з результатів, які зовсім відрізняються від того, що слід очікувати, то дані тренувань спрямовують алгоритм на правильний шлях.

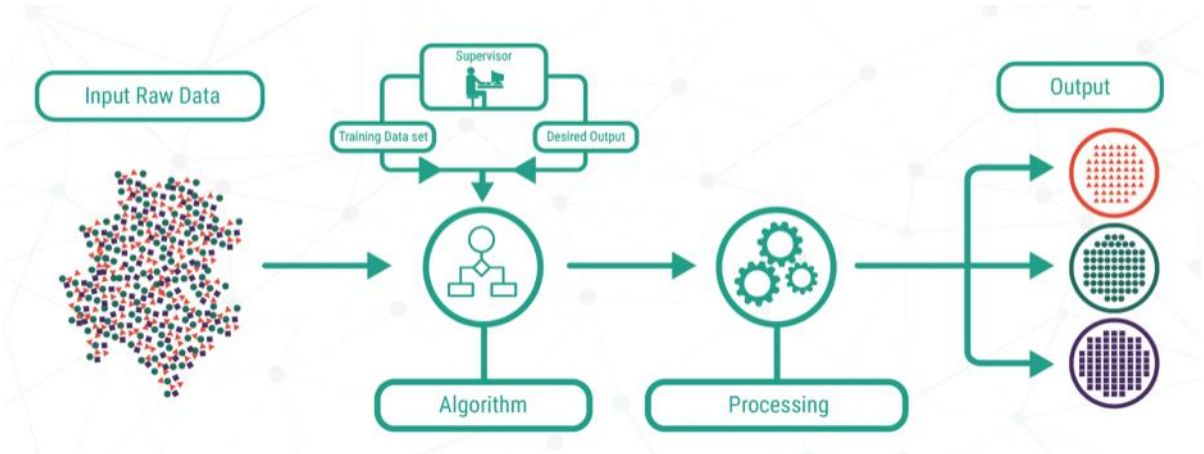

Рисунок 1.1 - Схема навчання з учителем [2]

Машинне навчання з учителем в даний час складає більшу частину машинного навчання, яке використовується системами у всьому світі. Вхідна змінна (*x*) використовується для з’єднання з вихідною змінною (*y*) за допомогою алгоритму. Вся

інформація, вихід, алгоритм та сценарій надаються людьми. Ми можемо зрозуміти контрольоване навчання ще краще, розглядаючи його через два види задач:

- Класифікація: Розпізнавання цифр є загальним прикладом вивчення класифікації. У загальному випадку, навчання класифікації підходить для будь-якої проблеми, коли результат класифікації є корисним і класифікацію легко визначити.

- Регресія: Проблеми, які можна класифікувати як регресійні проблеми, включають типи даних, в яких вихідні змінні встановлюються як дійсне число. Формат цієї проблеми часто випливає з лінійного формату.

Алгоритми навчання з учителем використовують набір даних, що містить ознаки, але кожен приклад також пов’язаний з міткою або ціллю. Наприклад, набір даних “Iris” анотований з різновидами ірису. Запропонований алгоритм навчання може вивчати набір даних “Iris” і навчитися класифікувати рослини ірису на три різних види на основі їх вимірювань.

1.1.2 Навчання без учителя

Незважаючи на те, що навчання без учителя ще не було реалізовано у більш широкому масштабі, ця методологія формує майбутнє машинного навчання та його можливостей. Ми завжди говоримо про машинне навчання, що дає необмежені можливості у майбутньому, але не можемо зрозуміти деталі. Кожного разу, коли люди говорять про комп’ютери та машини, що розвивають здатність “навчати себе” бездоганно, говорять про навчанням без учителя.

У процесі навчання без учителя система не має конкретних наборів даних, а результати більшості проблем значною мірою невідомі. У простій термінології система штучного інтелекту та об’єкт машинного навчання є сліпими, коли вони виконують операцію. Система має свої бездоганні та величезні логічні операції, спрямовані на це, але відсутність належних алгоритмів вводу та виводу робить процес ще більш складним. Навчання без учителя має можливість інтерпретувати та знаходити рішення для необмеженого обсягу даних за допомогою вхідних даних та

бінарного логічного механізму, присутнього у всіх комп’ютерних системах (рис. 1.2). Система взагалі не має розмічених даних.

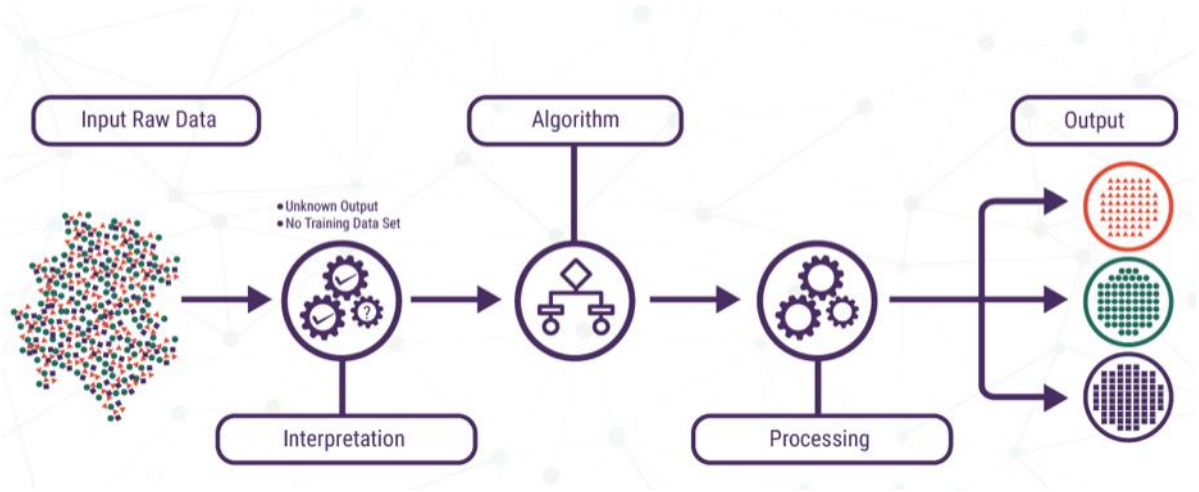

Рисунок 1.2 - Схема навчання без учителя [2]

Було б доречно зробити це розуміння ще більш простим, використовуючи приклад. Нехай є цифровий образ, на якому присутні різні кольорові геометричні фігури. Ці геометричні фігури повинні бути об’єднані в групи відповідно до кольорів та інших функцій класифікації. Для системи, яка працює через навчання з учителем, цей процес цілком простий. Процедура надзвичайно проста, тому що ви просто повинні навчити комп’ютер всім деталям, що відносяться до малюнків. Система може навчитись, що всі форми з чотирма сторонами відомі як квадрати, а інші з вісьмома сторонами називаються восьмикратними точками тощо. Система також може навчитись інтерпретувати кольори і “зрозуміти”, як класифікується світло.

Проте в процесі навчання без учителя весь процес стає трохи складнішим. Алгоритм процесу навчання без учителя має ті ж вхідні дані, що і для контрольованого аналога (у нашому випадку цифрові зображення, що відображають форми в різних кольорах).

Після того, як система має вхідні дані вона навчиться всьому, що можливо, за наявною інформацією. Справді, система сама по собі працює, щоб визначити проблему класифікації, а також різницю у формі та кольорах. З інформацією, пов’язаною з проблемою, яка знаходиться під рукою, система, що не контролюється,

потім розпізнає всі подібні об’єкти та об’єднує їх разом. Мітки, які вони надають цим об’єктам, будуть розроблені самою машиною. Технічно, можуть бути помилкові відповіді, оскільки існує певна ймовірність. Проте, так само як для людей, сила машинного навчання полягає у її спроможності розпізнавати помилки, вчитися на них і в кінцевому підсумку робити кращі оцінки наступного разу.

Алгоритми машинного навчання використовують набір даних, що містить багато ознак, а потім вивчають корисні властивості структури цього набору даних. У контексті глибокого навчання ми, як правило, хочемо вивчити весь розподіл імовірності, який генерує набір даних, як явно, так і в оцінці щільності або неявно для таких завдань, як синтез або зниження шуму (denoising). Деякі алгоритми навчання без учителя виконують інші ролі, такі як кластеризація, які полягають у розподілі набору даних на кластери подібних прикладів.

1.1.3 Навчання з підкріпленням

Навчання з підкріпленням - це ще одна галузь машинного навчання, яка набуває чималої корисності в тому, як це допомагає машині навчатися з її прогресу.

Навчання з підкріпленням - це навчання того, що робити - як співставити ситуації до дій, щоб максимізувати числовий сигнал нагороди (рис 1.3). Навчаючим алгоритмам не сказано, які дії потрібно виконувати, але, натомість, вони повинні визначити, які дії приносять найбільшу винагороду, намагаючись їх виконувати. У найцікавіших та складних випадках дії можуть впливати не тільки на негайну нагороду, а й на наступну ситуацію, а також на всі наступні нагороди. Ці дві характеристики - пошук проб і помилок і затримка нагороди - це дві найважливіших відмінних риси навчання з підкріпленням.

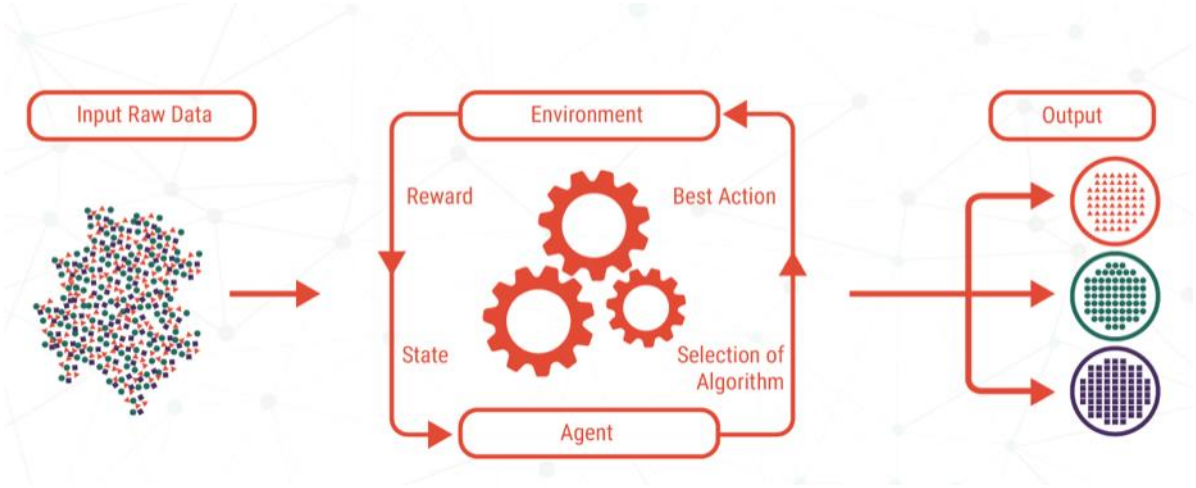

Рисунок 1.3 - Схема навчання з підкріпленням [2]

Навчання з підкріпленням відрізняється від навчання з учителем, який вивчається в більшості сучасних досліджень у галузі машинного навчання. Навчання з учителем - це навчальна програма з навчального набору позначених прикладів, наданих знайомим зовнішнім керівником. Кожен приклад - це опис ситуації разом із специфікацією - міткою правильної дії, яку система повинна прийняти до такої ситуації, яка часто визначає категорію, до якої належить ця ситуація. Об’єктом такого виду навчання є те, щоб система екстраполювала або узагальнювала свої відповіді, щоб вона правильно діяла в ситуаціях, які не існують у навчальному наборі. Це важливий вид навчання, однак він не є достатнім для навчання взаємодії. В інтерактивних завданнях часто неможливо отримати приклади бажаної поведінки, котра є правильною та є репрезентативною для всіх ситуацій, коли агент повинен діяти. На незрозумілих територіях, де можна очікувати, що навчання буде найбільш корисним для агента, треба вміти навчатись на власному досвіді.

Навчання з підкріпленням також відрізняється від того, що дослідники машинного навчання називають навчанням без учителя, що зазвичай полягає в пошуку структури, прихованої в наборах нерозмічених даних. Терміни “навчання з учителем” та “навчання без учителя” здавались б вичерпними, щоб класифікувати парадигми машинного навчання, але їх не достатньо. Незважаючи на те, що спонукання думати про навчання з підкріпленням як до навчання без учителя, оскільки воно не спирається на приклади правильної поведінки, навчання може

посилювати сигнал винагороди, а не намагатися знайти приховану структуру. Розкриття структури в досвіді агента, безумовно, може бути корисним для навчання з підкріпленням, але сама по собі не розглядає проблема навчання з підкріпленням, що полягає в максимізації сигналу винагороди. Тому можна вважати навчання з підкріпленням третьою парадигмою навчального процесу, поряд з навчанням з учителем та без учителя а також, можливо, іншими парадигмами.

Однією з проблем, що виникають у навчанні з підкріпленням а не в інших видах навчання, є компроміс між розвідкою та експлуатацією. Щоб отримати велику винагороду, агент навчання з підкріпленням повинен віддати перевагу діям, які він намагався в минулому, і виявив, що вони ефективні у виробництві винагороди. Але, щоб виявити такі дії, він повинен спробувати дії, які він раніше не обрав. Агент повинен використовувати той досвід, який він вже має, щоб отримати нагороду, але він також повинен навчитись, щоб в майбутньому робити кращі дії. Дилема полягає в тому, що ні дослідження, ні експлуатація не можуть бути здійснені винятково без усунення проблеми. Агент повинен спробувати різні дії та поступово віддавати перевагу тим, які йому видаються найкращими. По стохастичному завданню кожну дію слід багато разів повторити, намагаючись отримати достовірну оцінку її очікуваної нагороди. Розвідувально-експлуатаційну дилему інтенсивно вивчали математики протягом багатьох десятиліть, однак вона залишається невирішеною. Наразі ми просто відзначаємо, що вся проблема балансування розвідки та експлуатації навіть не виникає під час навчання з учителем та без учителя, принаймні у найпоширеніших формах.

Ще однією ключовою особливістю навчання з підкріпленням є те, що воно явно розглядає всю проблему цілеспрямованого агента, який взаємодіє з невизначеним середовищем. Це на відміну від багатьох підходів, які розглядають підзадачі, не розглядаючи, як вони можуть вписуватися в більшу картину. Наприклад, вже було згадано, що велика частина досліджень в галузі машинного навчання пов’язана з навчанням з учителем без чіткого визначення того, як нарешті така здатність буде корисна. Інші дослідники розробили теорії планування з загальними цілями, однак не беручи до уваги роль планування в процесі прийняття рішень у реальному часі або

питання про те, де будуть зроблені прогнозні моделі, необхідні для планування. Хоча ці підходи дали багато корисних результатів, їхнє зосередження на ізольованих підпроблемах є значним обмеженням.

Навчання з підкріпленням відбувається навпаки, починаючи з повного, інтерактивного агента пошуку цілей. Всі агенти навчання з підкріпленням мають чіткі цілі, можуть відчувати аспекти їх середовища та можуть вибирати дії, що впливають на їх середовище. Крім того, з самого початку зазвичай передбачається, що агент повинен працювати, незважаючи на значну невизначеність щодо середовища, з яким він стикається. Коли навчання з підкріпленням передбачає планування, воно повинно розглядати взаємодію між плануванням і вибором дій у режимі реального часу, а також питання, як вдосконалювати моделі навколишнього середовища. Коли навчання з підкріпленням включає навчання з учителем, це відбувається з певних причин, які визначають, які здібності є критичними та які не є. Для вивчення досліджень для досягнення прогресу важливі підзадачі повинні бути ізольованими та вивченими, але вони повинні бути підзадачами, які відіграють чітку роль у повних, інтерактивних агентах пошуку цілей, навіть якщо всі деталі повного агента ще не можуть бути заповнені.

Завдяки повному, інтерактивному агенту пошуку цілей ми не завжди маємо на увазі щось, що нагадує повний організм або робот. Це явний приклад, але повний, інтерактивний агент для пошуку цілей також може бути складовою частиною більшої системи, що веде себе. У цьому випадку агент безпосередньо взаємодіє з іншою більшою системою і побічно взаємодіє з середовищем більшої системи. Простий приклад - агент, який контролює рівень заряду акумулятора робота та надсилає команди до архітектури керування роботом. Навколишнє середовище цього агента - це робот разом із середовищем робота. Потрібно вийти за рамки найбільш очевидних прикладів агентів та їх оточення, щоб оцінити загальну структуру навчання з підкріпленням.

1.1.4 Напівавтоматичне навчання

У перших двох типах навчання або немає міток для всіх спостережень в наборі даних, або мітки присутні для всіх спостережень повністю. Напівавтоматичне навчання припадає між цими двома типами [3]. У багатьох практичних ситуаціях вартість розмітки даних досить висока, оскільки для цього потрібні кваліфіковані експерти (рис. 1.4). Отже, за відсутності розмітки в більшості спостережень, але за наявності невеликої кількості, алгоритми напівавтоматичного навчання є найкращими кандидатами для побудови моделі (рис. 1.5). Ці методи використовують ідею про те, що, хоча членство в групі нерозмічених даних невідоме, ці дані містять важливу інформацію про параметри групи.

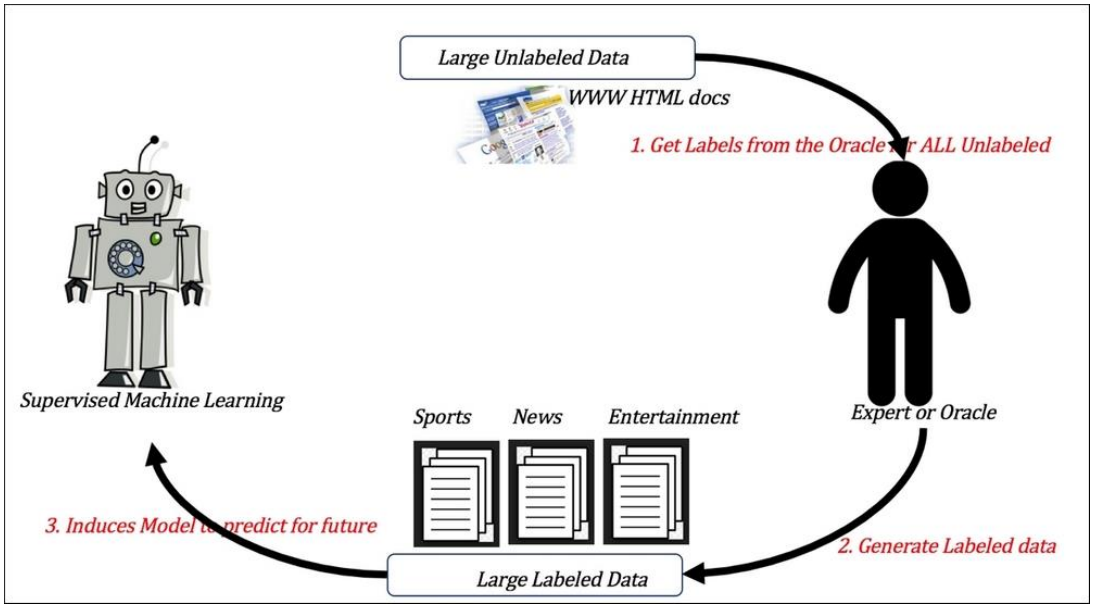

Рисунок 1.4 - Стандартний підхід до навчання з нерозміченими даними [3]

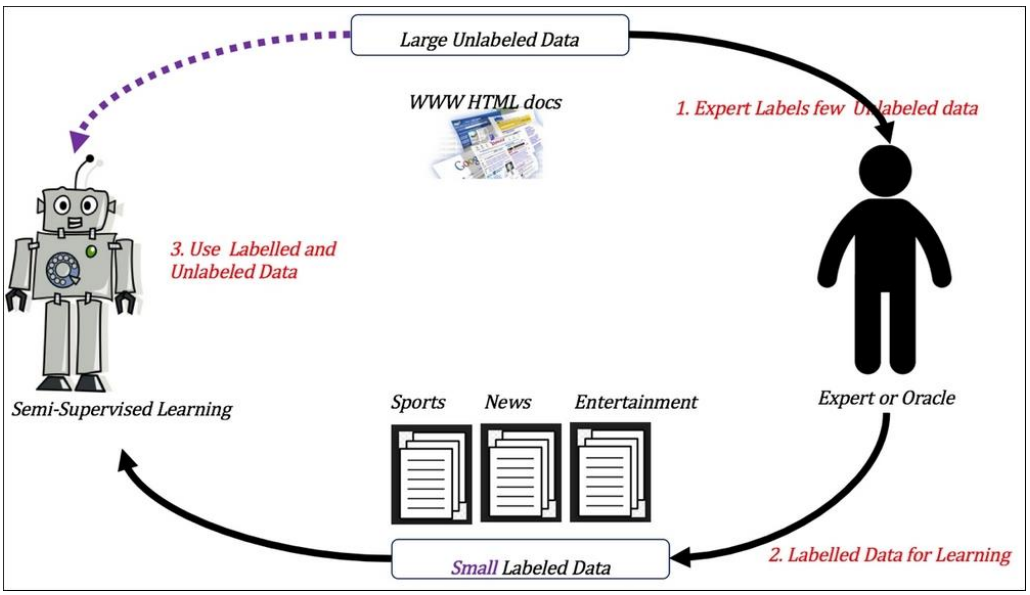

Рисунок 1.5 - Схема напівавтоматичного навчання [3]

У парадигмі напівавтоматичного навчання, для оцінки $P(y \mid x)$ або прогнозування y з x використовуються як незамічені приклади з $P(x)$ та позначені приклади з $P(x, y)$.

У контексті глибокого навчання під напівавтоматичним навчанням, як правило, розуміється вивчення представлення $h = f(x)$. Мета полягає в тому, щоб вивчити представлення так, щоб приклади з того самого класу мали однакові уявлення [4]. Навчання без учителя може дати корисні підказки для групування прикладів у просторі представлення. Приклади, які попадають в один кластер у вхідному просторі, повинні бути зіставлені з подібними уявленнями. Лінійний класифікатор у новому просторі може досягти кращого узагальнення у багатьох випадках [5, 6]. Довгий варіант такого підходу є застосування основних компонентів аналіз як етап попередньої обробки перед застосуванням класифікатора (за прогнозованими даними).

Замість того, щоб мати окремі компоненти з учителем та без учителя в моделі, можна побудувати модель, в яких генеративна модель $P(x)$ або $P(x, y)$ розділяє параметри з дискримінаційною моделлю $P(y \mid x)$. Тоді можна компрометувати контрольований критерій $-\log P(y \mid x)$ з неконтрольованою або генеративною (наприклад $-\log P(x)$ або $-\log P(x, y)$) моделлю. Потім генеративний критерій виражає особливу форму попередньої думки про рішення для проблеми навчання з

підкріпленням [7], а саме, що структура $P(x)$ пов’язана зі структурою $P(y \mid x)$ в спосіб, який схоплюється загальною параметризацією. Контролюючи, як велика частина генеративного критерію включена в сукупний критерій, можна знайти кращу компромісну ситуацію, ніж чисто генеративний чи чисто дискримінаційний критерій тренінгу [7, 8].

1.2 Штучні нейронні мережі

Штучна нейронна мережа (artificial neural network, ANN) - це математична модель, яка намагається імітувати структуру та функціональні можливості біологічних нейронних мереж [9]. Основним будівельним блоком кожної штучної нейронної мережі є штучний нейрон, тобто проста математична модель (функція). Така модель має три прості набори правил: множення, сумування та активація. На вході штучного нейрона входи зважуються, що означає, що кожне вхідне значення множиться з індивідуальною вагою. У середній частині штучного нейрона - сумарна функція, яка об’єднує всі зважені входи та зміщення. На виході з штучного нейрона сума попередньо зважених входів і зсуву відбувається через функцію активації, яка також називається передавальною функцією. Схема штучного нейрона зображена на рисунку 1.6.

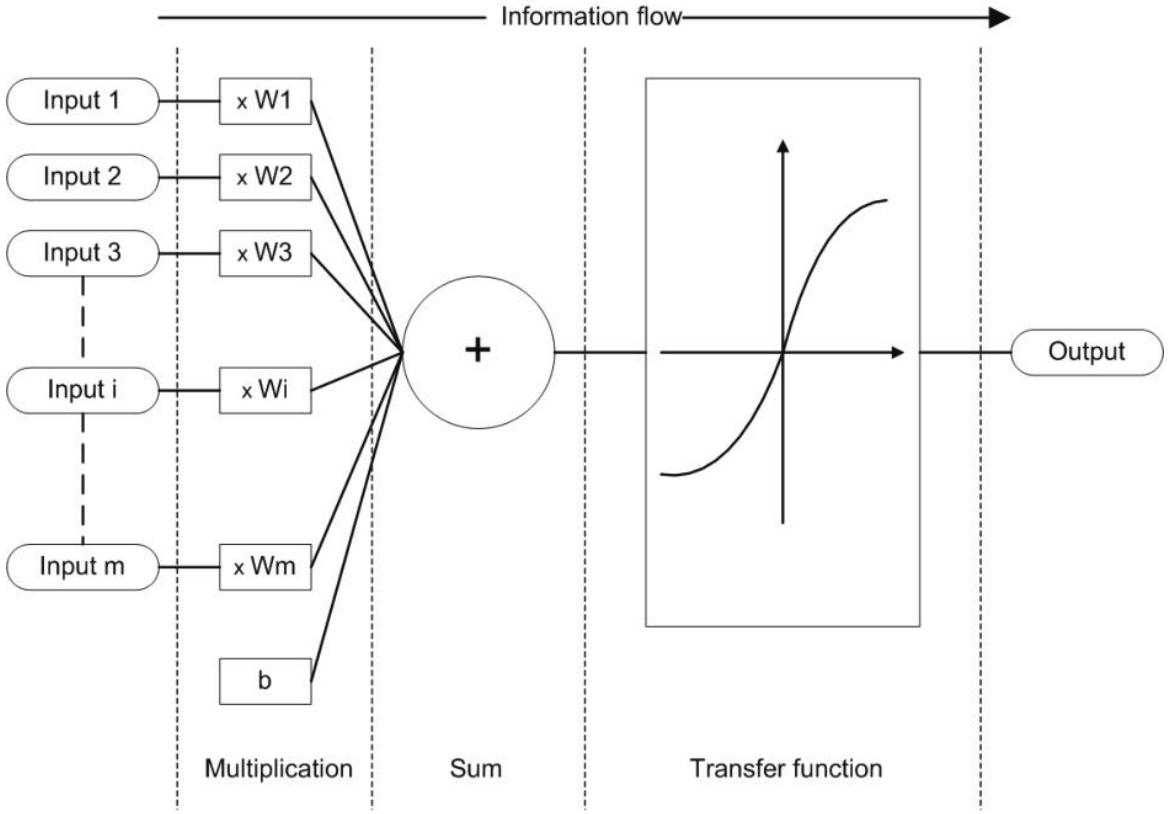

Рисунок 1.6 - Штучний нейрон [9]

Хоча робочі принципи та простий набір правил штучного нейрону виглядають як “нічого особливого”, повний потенціал та потужність розрахунків цих моделей проявляється, коли ми починаємо з’єднувати їх у штучні нейронні мережі (рис. 1.7). Ці штучні нейронні мережі використовують простий факт, що складність може вирости з декількох основних і простих правил.

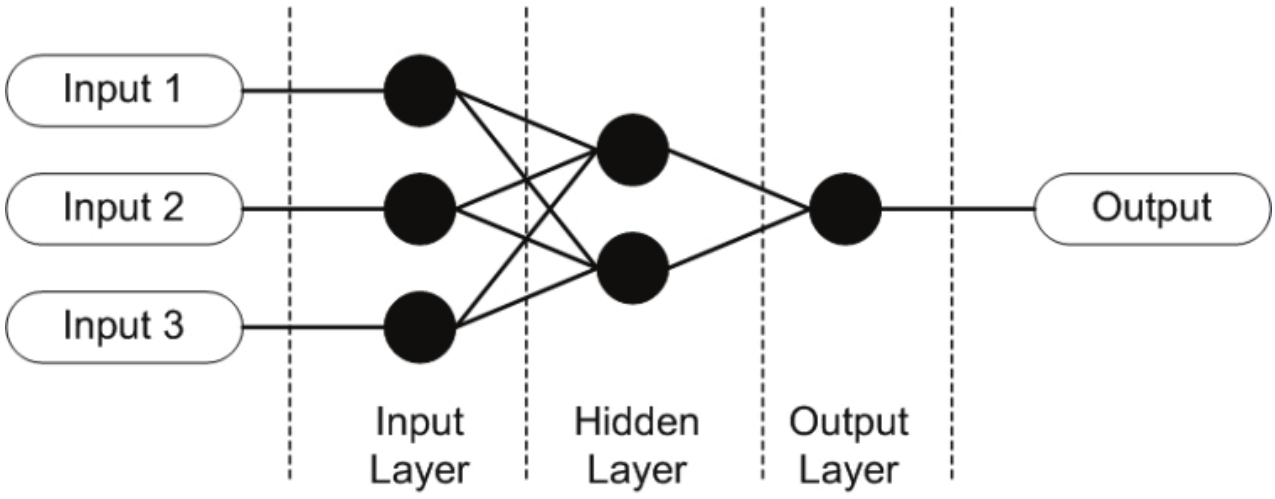

Рисунок 1.7 - Штучна нейронна мережа [9]

Для отримання повних переваг математичної складності, яка може бути досягнута через взаємозв’язок окремих штучних нейронів, а не просто роблячи систему складною та некерованою, ми зазвичай не сполучаємо ці штучні нейрони випадковим чином. У минулому дослідники виробили кілька стандартизованих топологій штучних нейронних мереж. Ці попередньо визначені топології допомагають простіше, швидше і ефективніше вирішити проблеми. Різні типи штучних нейронних мереж підходять для вирішення різних типів завдань. Визначивши тип заданої задачі, необхідно визначити топологію штучної нейронної мережі, яку ми збираємося використовувати, а потім налаштувати її. Нам потрібно точно налаштувати саму топологію та її параметри.

1.2.1 Штучний нейрон

Штучний нейрон є основним будівельним блоком кожної штучної нейронної мережі. Його конструкція та функціональність походять від спостереження біологічного нейрона, який є основним будівельним блоком біологічних нейронних мереж (систем), який включає головний мозок, спинний мозок та периферичні гангліїї. Подібності в дизайні та функціональних можливостях можна побачити на рисунку 1.8, де ліва частина фігури являє собою біологічний нейрон з його сомою, дендритами та аксоном, а правою частиною малюнка являє собою штучний нейрон з його входами, вагами, передавальною функцією, зміщенням і виходами.

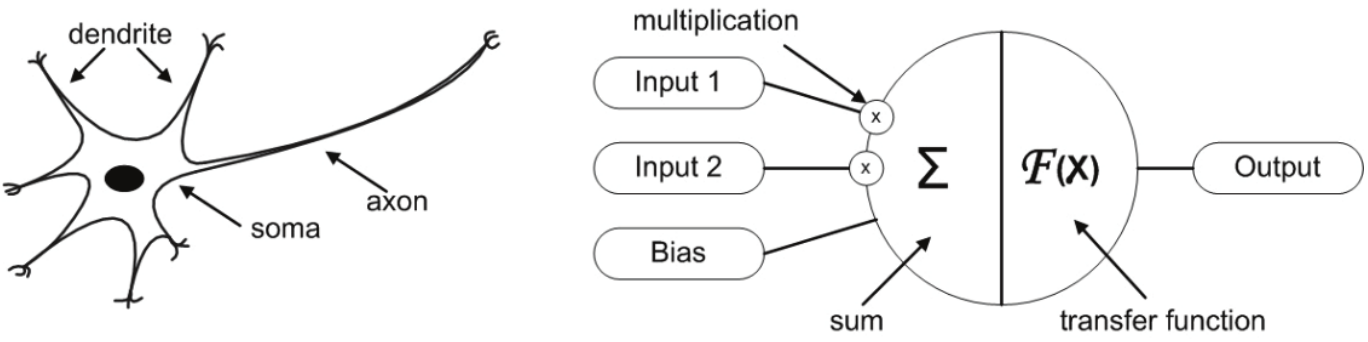

Рисунок 1.8 - Біологічний та штучний нейрони [9]

У випадку біологічного нейрона інформація потрапляє в нейрон через дендрит, сома обробляє інформацію і передає її через аксон. У випадку штучного нейрона інформація надходить у тіло штучного нейрона за допомогою вхідних даних, що зважуються (кожен вхід має бути індивідуально перемножений з вагою). Тіло штучного нейрона потім підсумовує зважені входи, зміщення та обробляє суму за допомогою передавальної функції. В кінці штучний нейрон передає оброблену інформацію за допомогою виходу(ів). Перевага простоти моделі штучного нейрона видно в її математичному описі, у формулі (1.1).

$$y = g\left(\sum_{i=0}^{n} w_i * x_i + b\right), \tag{1.1}$$

де y – вихід, g – функція активації, w – ваги, x – входи, b – зсув, n – кількість з'єднань.

Як видно з моделі штучного нейрона та її рівняння (1.1), основною невідомою змінною нашої моделі є його активаційна функція. Активаційна функція визначає властивості штучного нейрона і може бути будь-якою математичною функцією.

Деякі бажані властивості в функції активації включають:

Нелінійна. Коли функція активації нелінійна, то можна стверджувати, що дворівнева нейронна мережа є апроксиматором універсальної функції [10]. Функція активації ідентичності не задовольняє це властивість. Коли декілька шарів використовують функцію активації ідентичності, вся мережа еквівалентна одношаровій моделі. Спрощене пояснення приведене у рівняннях (1.2) - (1.5).

$$a^{(1)} = x \tag{1.2}$$

$$a^{(2)} = g\left(W^{(1)}a^{(1)} + b^{(1)}\right) \tag{1.3}$$

$$a^{(3)} = g\left(W^{(2)}a^{(2)} + b^{(2)}\right) \tag{1.4}$$

Якщо активаційна функція g лінійна, наприклад $g(x) = x$, то

$$a^{(3)} = W^{(2)}a^{(2)} + b^{(2)} = W^{(2)}\left(W^{(1)}a^{(1)} + b^{(1)}\right) + b^{(2)} = Wx + b \qquad (1.5)$$

Постійно диференційована - ця властивість бажана (rectify linear unit не є постійно диференційованою і має певні проблеми з оптимізацією на градієнтній основі) для введення методів оптимізації на основі градієнта. Функція активації двійкового кроку не є диференційованою на 0, і вона відрізняється від 0 для всіх інших значень, тому методи на основі градієнта не можуть досягти прогресу з ним [11].

Діапазон. Коли діапазон функції активації є обмеженим, методи тренувань на градієнті, як правило, стають більш стійкими, оскільки презентації шаблонів значно впливають лише на обмежені ваги. Якщо діапазон є нескінченним, тренінги, як правило, більш ефективні, оскільки представлення шаблонів значно впливають на більшість ваг. У другому випадку, зазвичай, необхідні менші темпи навчання.

Монотонна - коли функція активації є монотонною, поверхня помилки, пов’язана з одношаровою моделлю, гарантовано буде випуклою.

Гладкі функції з монотонною похідною - у деяких випадках вони довели, що вони краще узагальнюють. Аргумент на користь цих властивостей говорить про те, що такі функції активації більше відповідають бритві Оккама.

Основні активаційні функції та їх похідні представлені в таблиці 1.1.

1.2.2 Ваги

Нейрони зв’язані з іншими нейронами через ваги. В нейронній мережі інформація (знання) зберігається в вагах, а в процесі навчання нейронної мережі змінюються саме ваги. Вага є силою зв’язку, на скільки один нейрон впливає на інший; якщо збільшується вхідне значення, то на скільки впливає воно на вихідне. Вага, яка близька до нуля, означає, що зміна входу, не змінить виходу. Багато алгоритмів автоматично встановлюють ваги до нуля, щоб спростити мережу.

Негативні ваги означають, що збільшення входу зменшить вихід. Також потрібне зміщення (bias) - скільки виходу потрібно активізувати незалежно від входу.

Таблиця 1.1 - Активаційні функції

Назва	Рівняння	Похідна по x
Identity	$f(x) = x$	$f'(x) = 1$
Binary step	$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0, & x \neq 0 \\ ?, & x = 0 \end{cases}$
Logistic (Sigmoid)	$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH	$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified linear unit (ReLU)	$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$
Leaky ReLU	$f(x) = \begin{cases} 0.01x, & x < 0 \\ x, & x \geq 0 \end{cases}$	$f(x) = \begin{cases} 0.01, & x < 0 \\ 1, & x \geq 0 \end{cases}$

1.2.2.1 Ініціалізація

Навчання нейронної мережі відбувається через зміну ваг за допомогою алгоритмів оптимізації. Дані алгоритми працюють ефективніше, коли початкове значення ваг ініціалізоване певним чином.

Алгоритми навчання для глибоких навчальних моделей, як правило, є ітеративними за своїм характером і, таким чином, вимагають від користувача вказівки деякої початкової точки, з якої починати ітерації [4]. Крім того, тренування глибоких моделей є досить складним завданням, тому для більшості алгоритмів ініціалізація параметрів сильно впливає на результат. Початкова точка може визначити, чи взагалі сходиться алгоритм, при цьому деякі початкові точки настільки нестабільні, що алгоритм стикається з числовими труднощами і зовсім не працює. Коли навчання сходиться (converge), початкова точка може визначити, наскільки швидко навчання сходиться і чи сходить воно до точки з високою або низькою ціною (помилкою). Крім

того, точки зіставної вартості можуть мати різну узагальнену помилку, і початкова точка може вплинути на узагальнення.

Сучасні стратегії ініціалізації є простими та евристичними. Розробка вдосконалених стратегій ініціалізації є важким завданням, оскільки оптимізація нейронних мереж ще не зрозуміла. Більшість стратегій ініціалізації базуються на досягненні деяких хороших властивостей, коли мережа ініціалізується. Проте ми не розуміємо, які із цих властивостей зберігаються в тих обставин, після яких навчання починає діяти. Ще однією складністю є те, що деякі початкові моменти можуть бути корисними не лише з точки зору оптимізації, але з точки зору генералізації (узагальнення). Наше розуміння того, як початкова точка впливає на узагальнення, особливо примітивна, пропонуючи небагато інструкцій щодо вибору початкової точки.

Мабуть, єдиною властивістю, відомою з повною впевненістю, є те, що початкові параметри повинні “порушувати симетрію” між різними одиницями (нейронами). Якщо два приховані нейрона з однаковою функцією активації підключені до одного і того ж входу, то ці одиниці повинні мати різні початкові параметри. Якщо вони мають однакові початкові параметри, то алгоритм детерміністичного навчання, який застосовується до детерміністичних витрат і моделі, буде постійно оновлювати обидва ці одиниці таким же чином. Навіть якщо модель або алгоритм навчання спроможні використовувати стохастичність для обчислення різних оновлень для різних одиниць, зазвичай краще ініціалізувати кожну одиницю, щоб обчислити іншу функцію від усіх інших одиниць. Це може допомогти переконатися, що в нульовому просторі прямого розповсюдження відсутні шаблони вводу, і в нульовому просторі зворотного поширення не буде втрачено жодних градієнтних патернів. Метою того, що кожна одиниця обчислює іншу функцію, мотивує випадкову ініціалізацію параметрів. Ми могли б явно шукати великий набір базових функцій, які взаємно відрізняються один від одного, але це часто має помітну обчислювальну вартість. Наприклад, якщо у нас є якнайбільше виходів, як вхідних даних, ми можемо використовувати ортогоналізацію Грам-Шмідта на початковій матриці ваги і гарантувати, що кожна одиниця обчислює зовсім іншу функцію від

іншої одиниці. Випадкова ініціалізація з розподілу високої ентропії у багатомірному просторі є обчислювально дешевшою і навряд чи може привласнювати будь-які одиниці для обчислення однієї і тієї ж функції, як один до одного.

Як правило, ми встановлюємо зсуви для кожної одиниці для евристично обраних констант і випадково ініціалізуємо лише ваги. Додаткові параметри, наприклад, параметри, які кодують умовну дисперсію прогнозу, як правило, встановлюються на евристично обрані константи, так само як і зсув.

Ми майже завжди ініціалізуємо всі ваги в моделі до значень, обраних випадковим чином з гаусового або рівномірного розподілу. Вибір гаусового або рівномірного розподілу, мабуть, не має великого значення, але це ще не вивчено всебічно. Однак масштаб початкового розподілу має значний вплив як на результат процедури оптимізації, так і на здатність мережі узагальнювати.

Більші початкові ваги дадуть сильніший ефект розлому симетрії, що допоможе уникнути зайвих одиниць. Вони також допомагають уникнути втрати сигналу під час прямого або зворотного поширення через лінійну складову кожного шару більші значення в матриці, що призводить до збільшення виходів матричного множення. Однак початкові ваги, що занадто великі, можуть призвести до “вибуху” значень під час прямого поширення або зворотного поширення. У рекурентних мережах (recurrent neural network, RNN) великі ваги також можуть призвести до хаосу (така екстремальна чутливість до малих збурень вхідного сигналу, що поведінка детерміністичної процедури поширення вперед виявляється випадковою). До певної міри проблему градієнтного вибуху можна пом’якшити за допомогою градієнтного обрізання (порогові значення градієнтів перед виконанням етапу спуску градієнта). Великі ваги також можуть призвести до екстремальних значень, які викликають насиченість функції активації, що призводить до повної втрати градієнта через насичені одиниці. Ці конкуруючі фактори визначають ідеальну початкову шкалу ваг.

Деякі евристики доступні для вибору початкової шкали ваг. Один з евристичних засобів полягає в тому, щоб ініціалізувати ваги повністю підключеного шару з m входами та n виходами шляхом відбору кожного ваги з $U(-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}})$, тоді як

деякі вчені пропонують використовувати нормалізовану ініціалізацію, представлену в рівнянні (1.6).

$$W_{i,j} \sim U\left(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}}\right) \tag{1.6}$$

Цей останній евристичний метод призначений для компромісу між ціллю ініціалізації всіх шарів, щоб мати таку ж дисперсію активації, і ціллю ініціалізації всіх шарів мати однакову градієнтну дисперсію. Формула виведена, використовуючи припущення, що мережа складається лише з ланцюга множення матриць, без нелінійностей. Реальні нейронні мережі, очевидно, порушують це припущення, але багато стратегій, розроблених для лінійної моделі, працюють достатньо добре на своїх нелінійних аналогах.

Деякі вчені рекомендують ініціалізувати випадкові ортогональні матриці з ретельно вибраним масштабуванням або коефіцієнтом *k* коефіцієнта посилення, який пояснює нелінійність, застосовану на кожному шарі. Вони отримують конкретні значення коефіцієнта масштабування для різних типів нелінійних функцій активації. Ця схема ініціалізації також мотивована моделлю глибокої мережі як послідовності матричних множення без нелінійностей. За такою моделлю ця схема ініціалізації гарантує, що загальна кількість ітерацій для тренувань, необхідних для досягнення конвергенції, не залежить від глибини.

Збільшення масштабного коефіцієнта *k* підштовхує мережу до режиму, коли активація збільшується в нормі, коли вони поширюються вперед через мережу, а градієнти збільшують норму, коли вони поширюються назад. Правильне встановлення коефіцієнта посилення достатньо для навчання мереж глибиною до 1000 шарів без необхідності використовувати ортогональні ініціації. Ключовим уявленням про такий підхід є те, що в послідовних мережах активація та градієнти можуть зростати або зменшуватися на кожному кроці вперед чи назад, після випадкового поводження. Це відбувається тому, що мережі прямого поширення

використовують іншу матрицю ваги на кожному шарі. Якщо ця випадкова прогулянка налаштована на збереження норм, то передові мережі можуть в основному уникати проблеми “зникнення” і “вибуху” градієнтів, які виникають, коли на кожному кроці використовується однакова матриця ваги.

На жаль, ці оптимальні критерії для початкової ваги часто не призводять до оптимальної продуктивності. Це може бути з трьох різних причин. По-перше, ми можемо використовувати неправильні критерії - насправді не може бути корисним зберегти норму сигналу по всій мережі. По-друге, властивості, накладені при ініціалізації, можуть не зберегтися після початку навчання. По-третє, критерії можуть досягти успіху в підвищенні швидкості оптимізації, але випадково збільшують погрішність узагальнення. На практиці, як правило, потрібно розглядати шкалу ваг як гіперпараметр, оптимальне значення якого лежить десь приблизно поблизу, але не точно дорівнює теоретичним прогнозам.

Одним з недоліків правил масштабування, який встановлює всі початкові ваги для того самого стандартного відхилення, як, наприклад, $\frac{1}{\sqrt{m}}$, полягає в тому, що кожна індивідуальна вага стає надзвичайно малою, коли шари стають великими. Є альтернативна схема ініціалізації, яка називається розрідженою ініціалізацією, в якій кожна одиниця ініціалізується, щоб мати рівно *k* неоднакових ваг. Ідея полягає в тому, щоб зберегти загальну кількість вхідного сигналу в одиницю незалежно від кількості входів m, не роблячи величини одиничних вагових елементів *y* термінах *m*. Розріджена ініціалізація допомагає досягти більшої різноманітності серед одиниць під час ініціалізації. Однак, це також накладає дуже сильні попередні на вагах, які вибрані для отримання великих гаусових значень. Оскільки для градієнтного спуску потрібно значно скоротити “неправильні” великі значення, ця схема ініціалізації може спричинити проблеми для одиниць, таких як блоки максимуму (maxout), що мають кілька фільтрів, які повинні бути ретельно координуватися між собою.

Коли обчислювальні ресурси дозволяють це, як правило, доцільно розглядати початкову шкалу ваг для кожного шару як гіперпараметр і вибрати ці шкали за допомогою алгоритму пошуку гіперпараметрів, наприклад, випадковий пошук. Вибір того, чи слід використовувати густу або розріджену ініціалізацію, також може бути

гіперпараметром. З іншого боку, можна вручну шукати найкращі початкові шкали. Правильне правило для вибору початкових масштабів - це подивитися на діапазон або стандартне відхилення активації або градієнти на одиницю мініатюри даних. Якщо вага занадто мала, діапазон активації в міні-групі зменшиться, оскільки активації поширюються вперед через мережу. Багаторазово ідентифікуючи перший шар з неприйнятно малими активами та збільшуючи його ваги, можна в кінцевому рахунку отримати мережу з розумною початковою активацією протягом усього. Якщо навчання в цьому місці все ще занадто повільне, може бути корисно подивитися на діапазон або стандартне відхилення градієнтів, а також активацію. Ця процедура в принципі може бути автоматизованою і, як правило, менш обчислювальна, ніж оптимізація гіперпараметрів на основі помилки встановлення валідації, оскільки вона ґрунтується на зворотному зв’язку з поведінкою вихідної моделі на одній партії даних, а не на зворотній зв’язок з навченої моделі на набір перевірок. Протягом тривалого використання евристично цей протокол останнім часом був більш точно визначений і вивчений.

Підхід для встановлення зсувів має узгоджуватися з підходом до налаштування ваг. Встановлення зсувів в нуль сумісно з більшістю схем ініціалізації ваг.

Крім цих простих постійних чи випадкових методів ініціалізації параметрів моделі, можна ініціалізувати параметри моделі за допомогою машинного навчання. Обговорювана загальна стратегія полягає в тому, щоб ініціалізувати модель з учителем за допомогою параметрів, отриманих моделлю без учителя, підготовленою на тих самих ресурсах. Також можна виконувати навчання з учителем на схожому завданні. Навіть виконуючи навчання з учителем з незв’язаного завдання, іноді може прийти ініціалізація, що забезпечує більш швидку конвергенцію, ніж випадкова ініціалізація. Деякі з цих стратегій ініціалізації можуть призвести до швидшого зближення та кращого узагальнення, оскільки вони кодують інформацію про розподіл в початкових параметрах моделі. Інші, очевидно, добре працюють, перш за все, тому що вони встановлюють параметри для правильної шкали або встановлюють різні одиниці для обчислення різних функцій один від одного.

1.2.2.2 Візуалізація

Давайте візьмемо мережу, для того, щоб класифікувати рукописні цифри MNIST, та створимо сполучення безпосередньо від вхідного шару до вихідного шару без прихованих шарів між ними [12]. Таким чином, наша мережа виглядає так, як на рисунку 1.9.

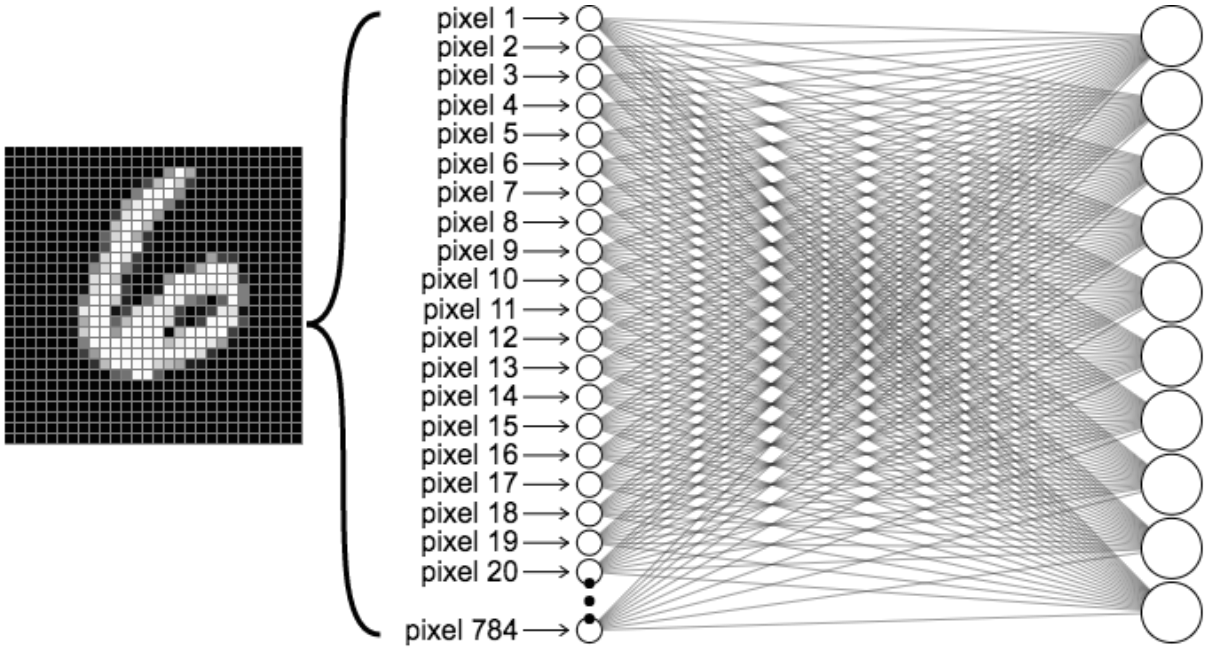

Рисунок 1.9 - Одношарова штучна нейронна мережа для розпізнавання рукописних цифр [12]

Згадаємо, коли вводять зображення в нашу нейронну мережу, ми візуалізуємо мережеву діаграму, “розгорнувши” пікселі в один стовпець нейронів, як показано на рисунку 1.10. Давайте зосередимо увагу лише на з’єднанні, підключеному до першого вихідного нейрона, який ми будемо позначати z, і позначатимемо кожен з вхідних нейронів та їхні відповідні ваги як x_i та w_i.

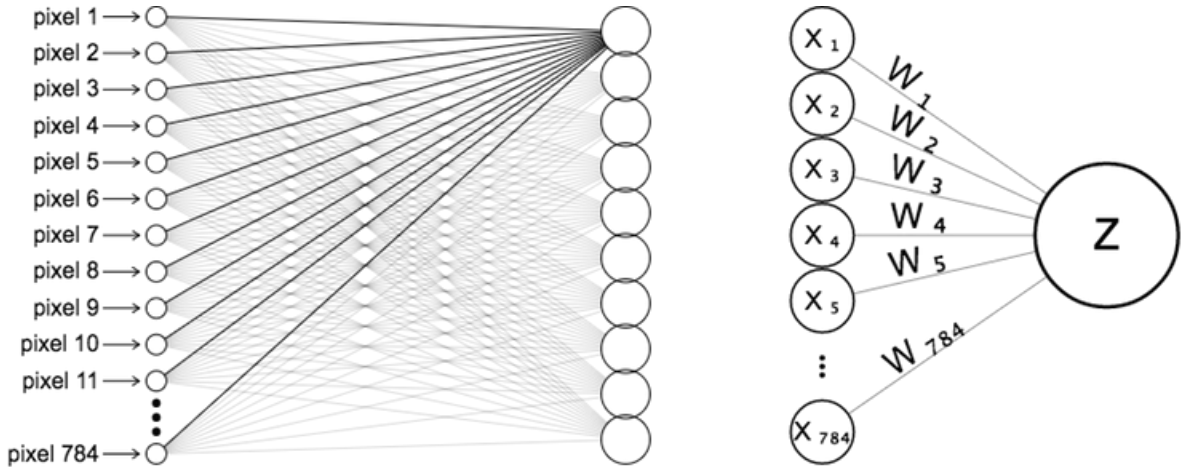

Рисунок 1.10 - З'єднання до вихідного нейрона [12]

Замість того, щоб розгортати пікселі, давайте розглянемо ваги як сітку розміром 28 × 28, де ваги розташовані точно так само, як їхні відповідні пікселі. Представлення наведене на попередньому рисунку виглядає інакше, ніж на рисунку 1.11, але вони виражають те саме рівняння, а саме $z = \sum wx + b$.

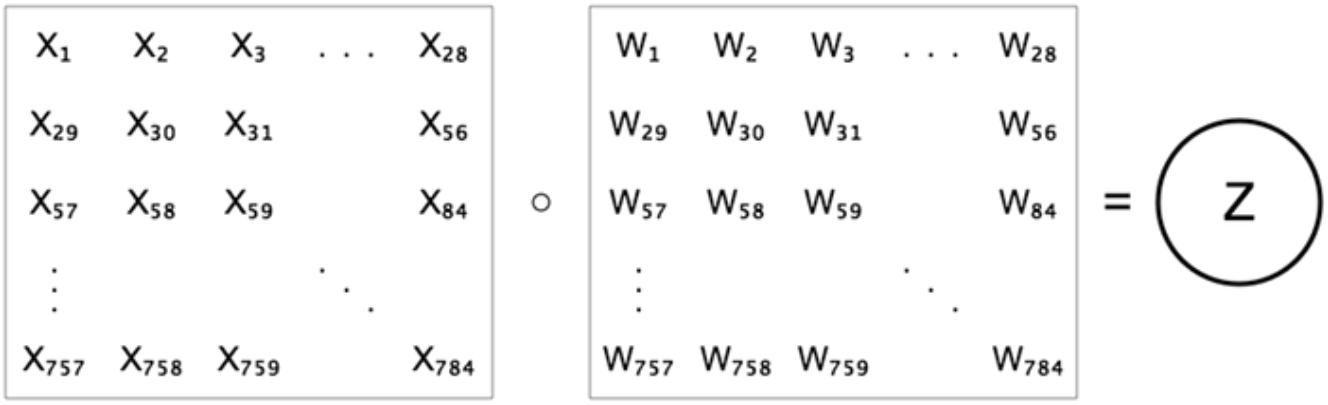

Рисунок 1.11 - Візуалізація множення пікселів на ваги [12]

Тепер давайте візьмемо натреновану нейронну мережу з цією архітектурою та візуалізуємо навчені ваги першого вихідного нейрона, який відповідає за класифікацію цифри нуль. Закодуємо колірним кодом, тобто щоб найнижче значення ваг було чорним, а найвище – білим (рис. 1.12).

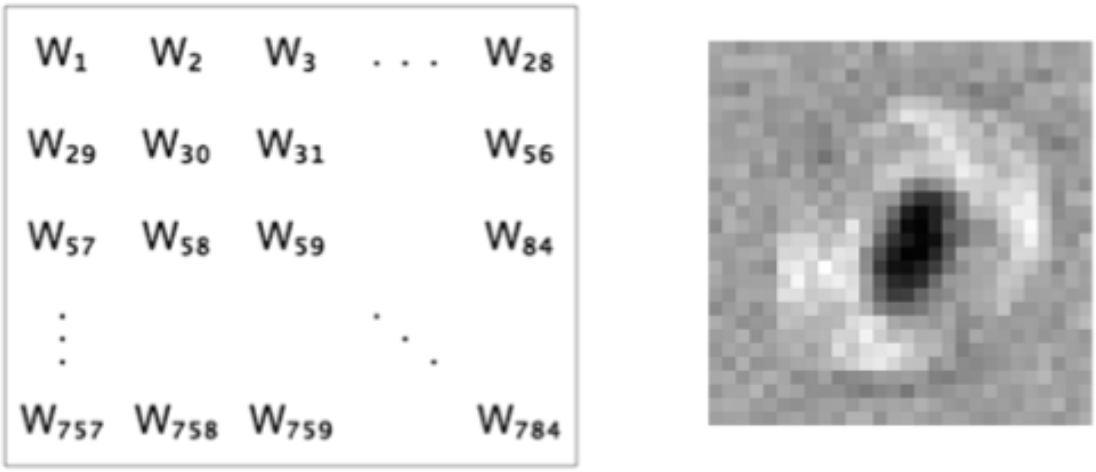

Рисунок 1.12 - Візуалізація ваг вихідного нейрона, який відповідає за класифікацію цифри 0 [12]

Нуль виглядає злегка розмитим. Причина, чому це виявляється таким чином, стає більш ясною, якщо ми думаємо про те, що робить нейрон. Оскільки він відповідає за класифікацію нулів, його метою є виведення високого значення для нулів та низького значення для не нулів. Він може отримати високі виходи для нуля, маючи великі ваги, вирівняні до пікселів, які, як правило, мають високе значення у зображеннях нулів. Одночасно він може отримати відносно низькі виходи для не нулів, маючи невеликі ваги, вирівняні до пікселів, які, як правило, високі у зображеннях, що не мають нуля, та низькі у зображеннях нулів. Зображення відносно чорного центру ваг залежить від того, що знімки нулів схильні вимикатися тут (отвір всередині нуля), але, як правило, вище для інших цифр.

Давайте подивимося на ваги, отримані для всіх 10 вихідних нейронів (рис. 1.13). Як очікувалося, всі вони виглядають дещо розмитими версіями наших десяти цифр. Вони з’являються майже так, якби ми усереднили багато зображень, що належать до кожного класу цифр.

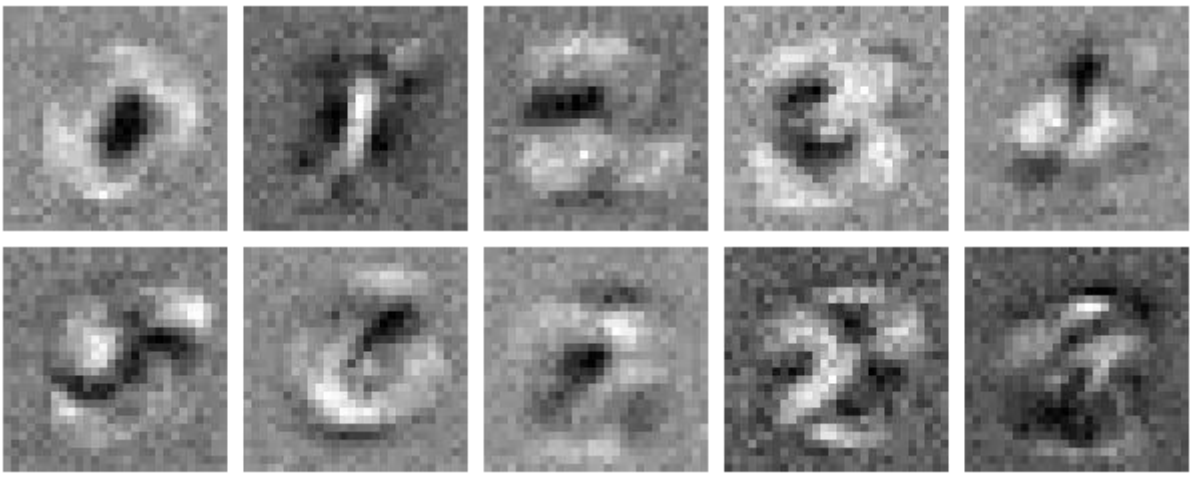

Рисунок 1.13 - Візуалізація ваг вихідних нейронів [12]

Припустимо, що ми отримуємо вхід із зображення “2”. Ми можемо передбачити, що нейрон, відповідальний за класифікацію двійок, повинен мати високе значення, оскільки його вага така, що висока вага, як правило, узгоджується з пікселями, які прагнуть бути високими в двійках. Для інших нейронів деякі ваги також будуть відповідати високим значенням пікселів, дещо вищі їх оцінки. Однак, набагато меншим буде перекриття, і багато хто з високими значеннями пікселів у цих зображеннях буде відхилятися за низькими вагами в інших нейронах. Функція активації не змінює цього, оскільки вона є монотонною стосовно входу, тобто, чим вище вхідний сигнал, тим вищий вихід.

Ми можемо інтерпретувати ці ваги як формування шаблонів вихідних класів. Це дійсно цікаво, тому що ми ніколи не вказували мережі заздалегідь про те, що це за цифри або що вони мають на увазі, але вони все одно нагадують ці класи об’єктів. Це натяк на те, що є особливо важливим всередині нейронних мереж: вони формують уявлення про об’єкти, на яких вони навчаються, і виявляється, ці уявлення можуть бути корисними для набагато більше, ніж проста класифікація чи прогнозування.

Це викликає набагато більше питань, ніж це дає відповіді, наприклад, що відбувається з вагами при додаванні прихованих шарів? Як приховані шари впливають на нашу нейронну мережу? Щоб побачити, спробуємо вставити прихований шар десяти нейронів у нашу мережу MNIST. Отже, наша нейронна мережа для класифікації рукописних цифр виглядає як показано на рисунку 1.14.

Наша проста метафора шаблону в 1-шаровій мережі вище не застосовується до даного випадку, тому що у нас більше немає 784 вхідних пікселів, які безпосередньо підключаються до класів виводу. У певному сенсі можна сказати, що ми “змусили” нашу оригінальну 1-шарову мережу вивчати ці шаблони, оскільки кожна з ваг пов’язана безпосередньо з міткою одного класу і, таким чином, лише вплинула на цей клас. Але в складній мережі, яку ми ввели зараз, ваги у прихованому шарі впливають на всі десять нейронів у вихідному шарі. Тож який вигляд цих ваг можна очікувати зараз?

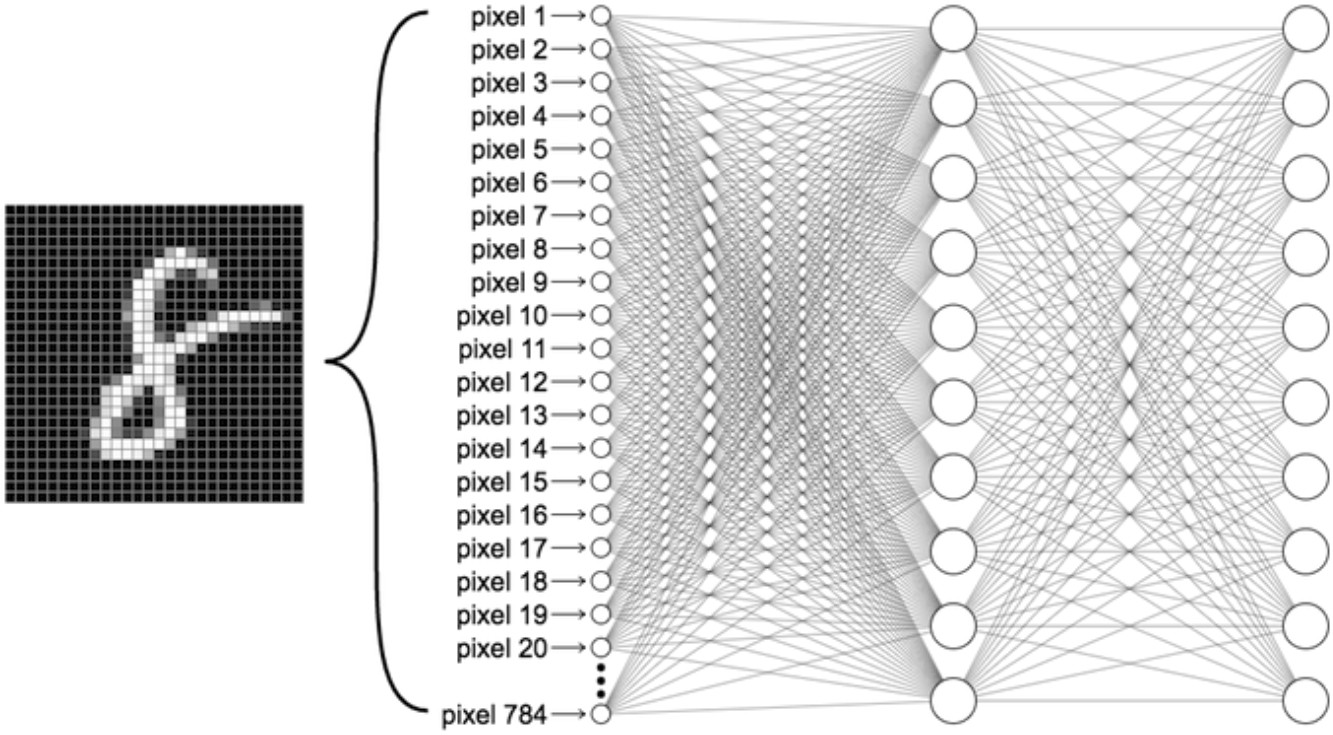

Рисунок 1.14 - Двошарова штучна нейронна мережа для розпізнавання рукописних цифр [12]

Щоб зрозуміти, що відбувається, ми будемо візуалізувати ваги в першому шарі, як і раніше, але ми також уважно подивимося, як їх активації потім об’єднуються в другий шар, щоб отримати оцінки класів. Нагадаємо, що зображення створить високу активацію в певному нейроні в першому шарі, якщо зображення в значній мірі “симпатизує” цьому фільтру. Таким чином, десять нейронів у прихованому шарі відображають присутність цих десяти функцій у вихідному зображенні. У вихідному

шарі один нейрон, що відповідає значенню класу, є вагою комбінацією з попередніх десяти прихованих активацій. Візуалізації ваг прихованого і вихідного шарів та активації нейронів зображені на рисунках 1.15 та 1.16.

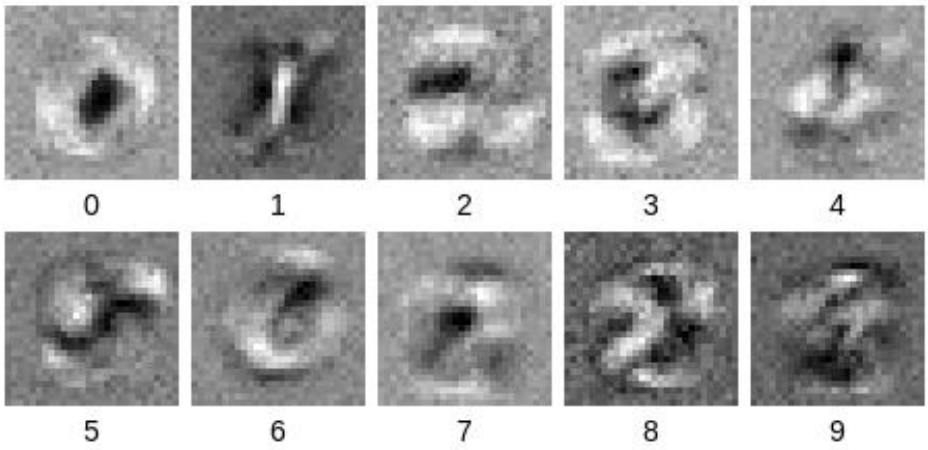

Рисунок 1.15 - Візуалізація ваг першого (прихованого) шару нейронів [12]

Почнемо з перших шарів ваг, візуалізованих у верхній частині. Вони більше не виглядають як шаблони класу зображень, але більше незнайомі. Деякі з них виглядають як псевдо цифрами, а інші, як компонентами цифр: половина петель, діагональні лінії, дірки тощо.

Рядки нижче зображень фільтра відповідають нашим вихідним нейронам, по одному для кожного класу зображень. Стовпчики означають ваги, пов’язані з кожним із десяти активацій фільтрів із прихованого шару. Наприклад, клас “0”, підтримують фільтри першого рівня, які є високими вздовж зовнішнього контура (коли нульова цифра має тенденцію з’являтися). Це пригнічує фільтри, де високі пікселі в середині (де зазвичай зустрічаються отвори в нулі). Клас “1” майже протилежне цьому, надаючи перевагу фільтрам, які є сильними в середині, де ви можете бачити вертикальну полосу одиниці.

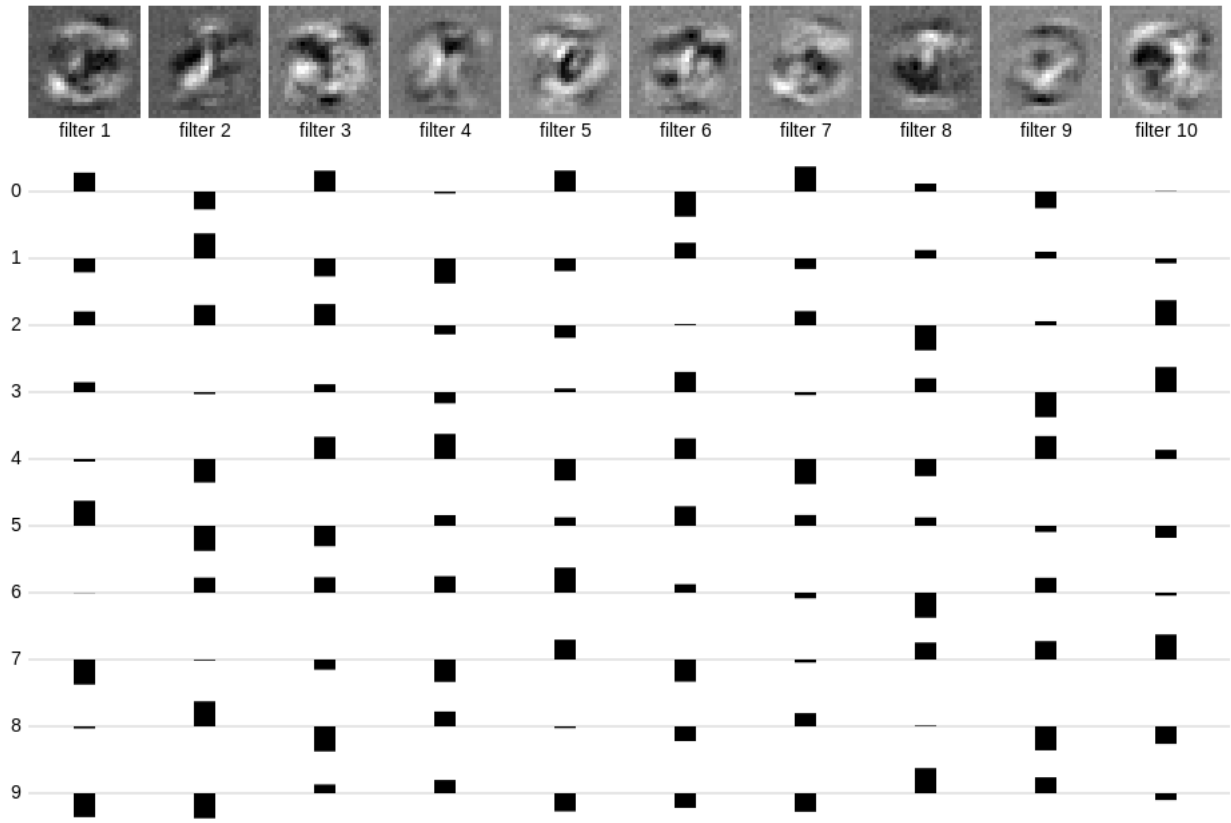

Рисунок 1.16 - Візуалізація ваг другого (вихідного) шару нейронів та активації нейронів для кожного класу [12]

Перевагою такого підходу є гнучкість. Для кожного класу є більш широкий спектр вхідних візерунків, які стимулюють відповідний вихідний нейрон. Кожен клас може бути викликаний наявністю декількох абстрактних функцій з попереднього прихованого шару або їх комбінації. По суті, ми можемо вивчати різні типи нулів, різних типів і т. д. Для кожного класу. Це зазвичай, але не завжди, покращить роботу мережі для більшості завдань.

У одношарових і багатошарових нейронних мережах кожен шар має подібну функцію; він перетворює дані з попереднього шару на “вищий рівень” представлення цих даних. Під “вищим рівнем” ми маємо на увазі, що в ньому міститься компактне та більш виразне уявлення про ці дані, так як підсумок є “високим рівнем” репрезентації. Наприклад, у вищезазначеній 2-х шаровій мережі ми перенесли пікселі низького рівня на високорівневі ознаки, знайдені в цифрах (штрихи, цикли тощо) у першому шарі, а потім нанесені на них ті високопоставлені функції в вищий рівня

представлення в наступному шарі. Це поняття перетворення даних у меншу, але більш значущу інформацію є основою машинного навчання та первинної здатності нейронних мереж.

Додавши прихований шар в нейронну мережу, ми даємо йому можливість вивчати функції на різних рівнях абстракції. Це дає нам багате представлення даних, в яких ми маємо ознаки низького рівня в ранніх шарах, а також ознак високого рівня в пізніших шарах, які складаються з попередніх ознак шару.

Як ми бачили, приховані шари можуть підвищити точність, але лише в обмеженій мірі. Додавання все більше і більше шарів швидко припиняє покращувати точність і приходить до обчислювальної вартості - ми не можемо просто попросити нашу нейронну мережу запам’ятовувати всі можливі версії класу зображень через її приховані шари.

1.2.3 Структура

При об’єднанні двох або більше штучних нейронів ми отримуємо штучну нейронну мережу. Якщо єдиний штучний нейрон практично не має користі у вирішенні реальних проблем, то штучні нейронні мережі мають. Справді, штучні нейронні мережі здатні вирішувати складні реальні проблеми, обробляючи інформацію в своїх основних будівельних блоках (штучних нейронах) нелінійно, розподіленою, паралельно та локально. Те, як окремі штучні нейрони взаємопов’язані, називаються топологією, архітектурою або графом штучної нейронної мережі.

1.2.3.1 Шари

Найбільш основним способом з’єднання нейронів з утворенням графів є сполучення всього з абсолютно всім [13]. Це видно в мережах Хопфілда та машинах Больцмана. Звичайно, це означає, що кількість зв’язків зростає експоненційно, але виразність безкомпромісна. Це називається повністю зв’язаними мережами.

Через деякий час було виявлено, що розбиття мережі на окремі шари є корисною функцією, де визначення шару являє собою набір або групу нейронів, які не пов’язані один з одним, але лише нейронам з інших груп. Це поняття, наприклад, використовується в обмежених машинах Больцмана (Restricted Boltzmann Machine, RBM). Ідея використання шарів в даний час узагальнюється для будь-якої кількості шарів і її можна знайти практично в усіх поточних архітектурах. Це (можливо, заплутано) також називають повністю зв’язними або повністю пов’язаними, тому що фактично повністю з’єднані мережі досить незвичні.

Згорткові шари (convolutionally connected layers) стають ще більш обмеженими, ніж повністю пов’язані шари: ми підключаємо кожний нейрон тільки до нейронів в інших групах, що знаходяться поруч. Зображення та звукові хвилі містять дуже велику кількість інформації, якщо вони використовуються для безпосереднього передавання інформації один одному в мережу (наприклад, використання одного нейрона на піксель). Ідея згорткових зв’язків полягає в тому, що просторова інформація, мабуть, важлива для збереження. Виявилося, що це гарна порада, оскільки вона використовується у багатьох нейронних мережевих додатках, що базуються на зображеннях та звукових хвилях. Однак ця установка менш виразна, ніж повністю пов’язані шари. По суті це є способом “важливості” фільтрації, вирішуючи, які важко згруповані інформаційні пакети; згорткові з’єднання великі для зменшення розмірності. На якій просторовій відстані нейрони можуть все ще бути зв’язаними, залежить від реалізації, але діапазони вище, ніж 4 або 5 нейронів рідко використовуються. Зауважте, що “просторовий часто” відноситься до двовимірного простору, тому більшість уявлень показують, що тривимірні шари нейронів пов’язані; діапазон з’єднання застосовується у всіх вимірах.

Інший варіант - це, звичайно, випадково пов’язані нейрони. Це також відбувається в двох основних варіантах: дозволяючи певний відсоток усіх можливих з’єднань або з’єднати певний відсоток нейронів між шарами. Випадкові підключення допомагають лінійно зменшувати продуктивність мережі і можуть бути корисними у великих мережах, де повністю пов’язані шари виникають із проблемами продуктивності. Рідше пов’язаний шар з трохи більшою кількістю нейронів може

краще працювати в деяких випадках, особливо там, де потрібно зберігати велику кількість інформації, але не потрібно обмінюватися такою кількістю інформації (трохи схожою на ефективність шарів, які потім рандомізовані). Також використовуються дуже рідко пов’язані системи (1 або 2). Тим більше, що у випадку зі спайковими мережами це має великий сенс, тому що чим більше зв’язків має нейрон, тим менше енергії кожна вага буде переносити, тобто менш поширювані та повторювані патерни.

З’єднання з затримкою в часі (time delayed connections) - це зв’язки між нейронами (часто з одного шару і навіть пов’язані з собою), які не отримують інформацію з попереднього шару, але з шару з минулого (попередня ітерація, в основному). Це дозволяє зберігати зв’язану з часом (час, послідовність або порядок) інформацію. Ці типи з’єднань час від часу часто змінюються вручну, щоб очистити “стан” мережі. Основною відмінністю від регулярних з’єднань є те, що ці з’єднання постійно змінюються, навіть якщо мережа не проходить навчання.

1.2.3.2 Приклади структур нейронних мереж

Нейронна мережа прямого поширення (feed forward neural networks, FFNN) і персептони подають інформацію з передньої до задньої (вхід і вихід, відповідно). Нейронні мережі часто описуються як такі, що мають шари, де кожен шар складається з паралельно вхідних, прихованих або вихідних комірок. Шар сам по собі не має зв’язків і загалом два сусідніх шару повністю пов’язані (кожен нейрон утворює один шар для кожного нейрона до іншого шару). Найпростіша дещо практична мережа має два вхідні нейрона та однин вихідний, які можна використовувати для моделювання логічних воріт (рис. 1.17). Один, як правило, готує FFNN через зворотне розповсюдження, надаючи мережеві набори даних про те, “що відбувається” та “що ми хочемо отримати”. Це називається контрольоване навчання, на відміну від безконтрольного навчання, де ми лише даємо йому вхід і дозволяємо мережі заповнити прогалини. Помилка при зворотному поширенні часто є деякою зміною різниці між входом і виходом (наприклад, середня квадратна помилка або просто

лінійна різниця). Враховуючи, що мережа має достатньо прихованих нейронів, вона теоретично завжди може моделювати зв’язок між входом і виходом. Практично їх використання набагато обмежені, але вони широко поєднуються з іншими мережами для формування нових мереж.

Рисунок 1.17 – Структура нейронної мережі прямого поширення [14]

Мережа Хопфілда (Hopfield network, HN) - це мережа, де кожен нейрон підключається до кожного іншого нейрону; це повністю заплутана “тарілка спагеті”, оскільки навіть всі вузли функціонують як все (рис. 1.18). Кожен вузол вводиться перед тренуванням, потім приховано під час тренувань та виводу після нього. Мережі навчаються, встановлюючи значення нейронів до бажаного малюнка, після чого ваги можуть бути обчислені. Ваги після цього не змінюються. Після навчання для одної або декількох моделей, мережа завжди збігатиметься з одною із вивчених моделей, оскільки мережа стабільна лише в тих станах. Зауважте, що це не завжди відповідає бажаному стану. Частково стабілізується через загальну “енергію” або “температуру” мережі, що поступово зменшується під час навчання. Кожен нейрон має поріг активації, який дорівнює цій температурі, яка, якщо перевищити сумарний вхід, змушує нейрон приймати форму одного з двох станів (зазвичай -1 або 1, іноді 0 або 1). Оновлення мережі може виконуватися синхронно або частіше один за одним. Якщо оновлено один за одним, то створюється чесна випадкова послідовність для того, щоб організувати, які ярлики оновлювати в якому порядку (справедливий випадковий всі параметри (n), що відбуваються рівно раз на кожні n елементів). Таким чином, ви можете визначити, коли мережа стабільна (виконується конвергенція), коли кожен нейрон був оновлений, і жодний з них не був змінений, мережа стабільна

(annealed). Ці мережі часто називають асоціативною пам’яттю, оскільки вони сходяться до найбільш схожих станів як вхідних; якщо люди бачать половину столу, то ми можемо зобразити іншу половину.

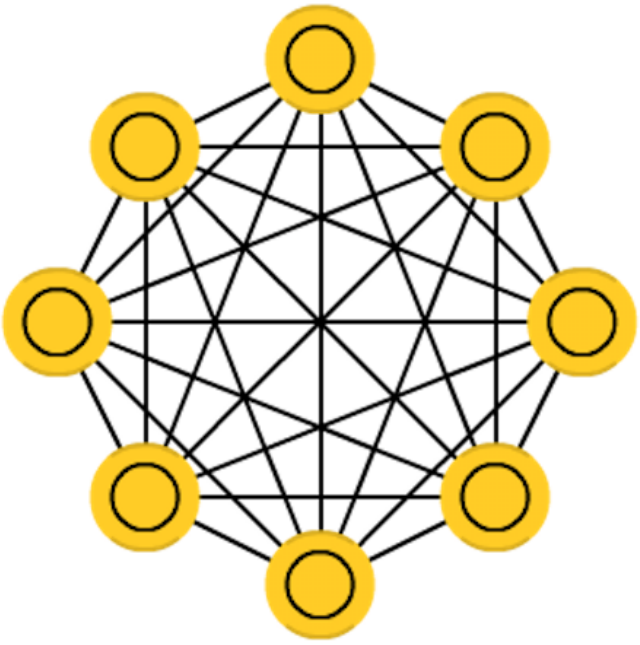

Рисунок 1.18 - Структура мережі Хопфілда [14]

Машини Больцмана (Boltzman Machine, BM) дуже схожі на HN, але: деякі нейрони відмічені як вхідні нейрони, а інші залишаються “прихованими” (рис. 1.19). Вхідні нейрони стають вихідними нейронами в кінці повного оновлення мережі. Воно починається з довільних ваг і навчається через зворотне поширення, або останнім часом через контрастну дивергенцію (ланцюги Маркова використовується для визначення градієнтів між двома інформаційними досягненнями). У порівнянні з HN, нейрони більшою мірою мають двійкові візуальні активації. Машини Больцмана є стохастичними мережами. Процес тренувань та виконання BM досить схожий на HN: він встановлює вхідні нейрони на певні закріплені значення, після чого мережа встановлюється довільно. Хоча вільні клітини можуть отримати будь-яке значення, і ми повторюємось назад і вперед між входом і прихованими нейронами. Активація контролюється глобальним значенням температури, що при зниженні знижує енергію

клітин. Ця нижча енергія призводить до стабілізації їхніх моделей активації. Мережа досягає рівноваги з урахуванням потрібної температури.

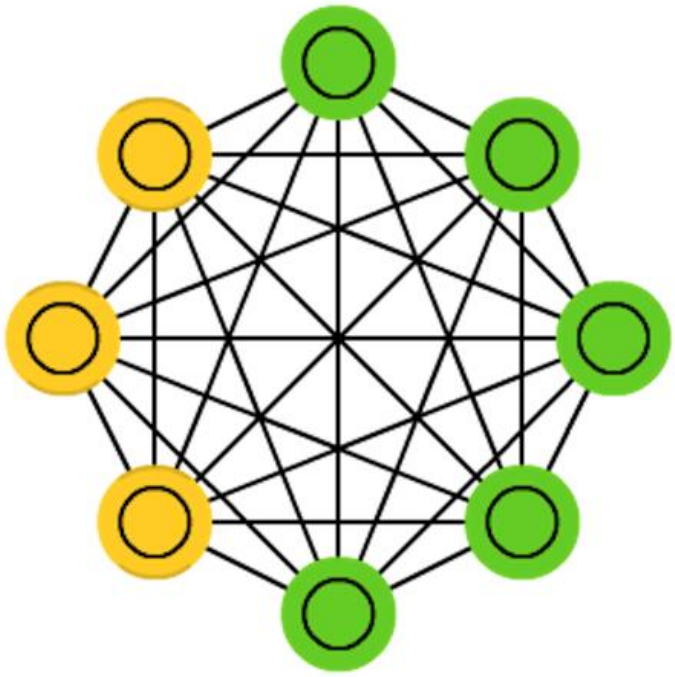

Рисунок 1.19 - Структура машини Больцмана [14]

Автокодери (Autoencoders, AE) дещо схожі на FFNNs, оскільки AE більше схожий на модифіковану FFNN, ніж принципово іншу архітектура (рис. 1.20). Основна ідея автокодерів полягає в тому, щоб автоматично кодувати інформацію (як у компресії, а не шифруванні), звідси, таке ім’я. Вся мережа завжди нагадує пісочний годинник, за формою, з меншими прихованими шарами, ніж вхідні та вихідні шари. AE завжди симетричні навколо середнього шару(-ів) (один або два залежно від рівномірного або непарного кількості шарів). Найменший шар(и) майже завжди знаходиться в середині, місце, де інформація найбільш стиснута. Все до середини називається частиною кодування, все після середини декодування а середина - кодом. Можна навчити їх, використовуючи зворотне поширення під час подачі вхідного сигналу, і встановити помилку як різницю між входом і тим, що вийшло. AE можуть бути побудовані симетрично, коли мова йде також про ваги, тому ваги кодування такі самі, як ваги для декодування.

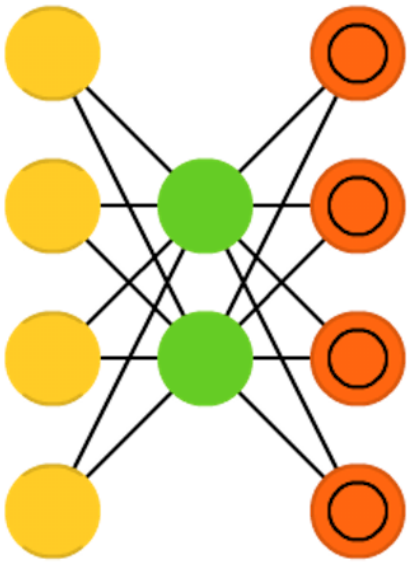

Рисунок 1.20 - Структура автоенкодера [14]

Згорткові нейронні мережі (convolutional neural networks, CNN) сильно відрізняються від більшості інших мереж. Вони в основному використовуються для обробки зображень, але можуть також використовуватися для інших типів входів, таких як аудіо. Типовим випадком використання для CNN є те, де мережі передаються на вхід зображення, і мережа класифікує дані: вона виводить «кіт», якщо їй було передано зображення коті та «собака», якщо їй було передано зображення собаки. CNN, як правило, на вході мають «сканера», який не призначений для аналізу всіх навчальних даних одночасно. Наприклад, щоб передати на вхід зображення розміром 200 x 200 пікселів, не потрібно створювати шар з 40 000 вузлів. Краще створити вхідний шар, що буде сканувати, розміром 20 x 20, який буде сприймати 20 x 20 пікселів зображення (зазвичай, починаючи з лівого верхнього кута). Як тільки дані пройшли через вхід (і можливо, ці дані були використаний для тренування), вхідний шар приймає наступні 20 x 20 пікселів: сканер переміщується на один піксель праворуч. При цьому зображення не розбивається на блоки розміром 20 x 20, а сканер ковзає по цьому зображенню із заданим кроком. Ці вхідні дані потім подаються на вхід до згорткових шарів замість звичайних шарів, де не всі вузли з’єднуються з усіма вузлами. Кожен вузол пов'язаний лише з найближчими вузлами (наскільки близько залежить від реалізації, але зазвичай не більше кількох вузлів). Ці згорткові шари, як

правило, зменшуються, коли вони стають більш глибокими (таким чином, 20, ймовірно, йтиме до шару 10, а потім за шаром 5). Степінь двійки дуже часто використовуються тут: 32, 16, 8, 4, 2, 1. Крім цих згорткових шарів, згорткові нейронні мережі також часто мають pooling (накопичення) шари. Pooling - це спосіб фільтрації деталей: загальноприйнята методика pooling - максимальне об'єднання (max poolig), де ми приймаємо 2 × 2 пікселі і пропускаємо далі піксель із найбільшою кількістю червоного кольору. Щоб застосувати CNNs для аудіо, подаються вхідні звукові хвилі, сегмент за сегментом. У реальних реалізаціях CNN часто приєднають до кінця FFNN для подальшої обробки даних. Ці мережі називаються DCNN, але імена для цих мереж часто використовуються взаємозамінно.

1.2.4 Навчання та оптимізація

Процедура, що використовується для виконання навчального процесу в нейронній мережі, називається алгоритмом навчання [15]. Існує багато різних алгоритмів навчання, що мають різні характеристики та продуктивність.

Розглянемо проблему навчання нейронних мереж з учителем. Проблема навчання в нейронних мережах сформульована з точки зору мінімізації функції втрат, *f*. Ця функція в цілому складається з помилки та умов регуляризації. Поняття про помилку оцінює, як нейронна мережа підходить до набору даних. З іншого боку, термін регуляризації використовується для запобігання перенавчання шляхом контролю ефективної складності нейронної мережі.

Функція втрат залежить від адаптивних параметрів (відхилень та синаптичних ваг) в нейронній мережі. Ми можемо зручно об'єднати їх у єдиний *n*-мірний ваговий вектор *w*. На рисунку 1.21 представлена функція втрат *f (w).*

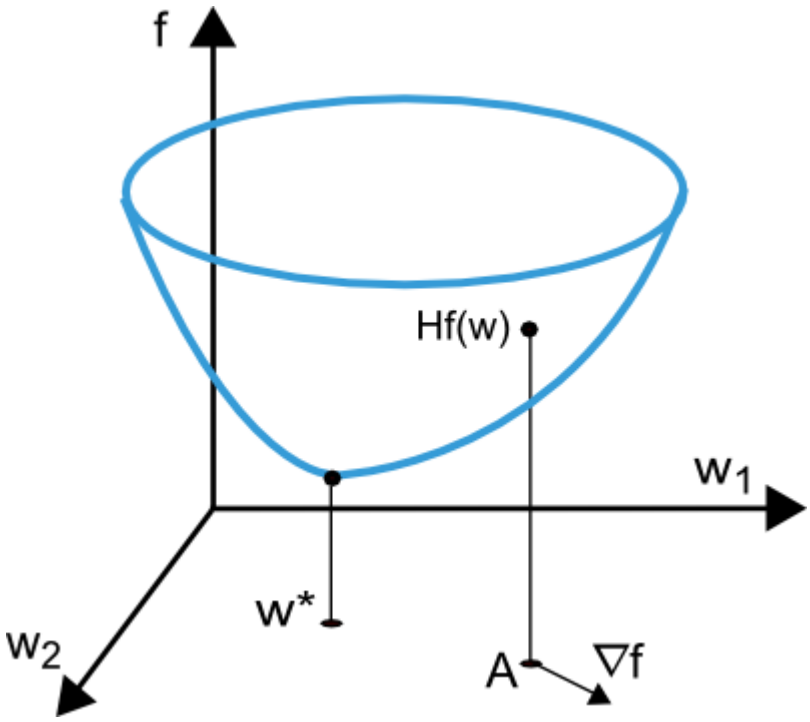

Рисунок 1.21 - Візуалізація функції витрат [15]

Як ми бачимо на попередньому знімку, точка w^* - мінімум функції втрат. У будь-якій точці A ми можемо обчислити першу та другу похідні функції втрат. Перші похідні згруповані у градієнтному векторі як у рівнянні (1.7).

$$\nabla_i f(w) = \frac{df}{dw_i} \tag{1.7}$$

Аналогічно, друга похідна від функції втрат можна згрупувати в матрицю Гессіана за рівнянням (1.8).

$$H_{i,j} f(w) = \frac{d^2 f}{dw_i dw_i} \tag{1.8}$$

Проблема мінімізації безперервних і диференційованих функцій багатьох змінних була широко вивчена. Багато загальноприйнятих підходів до цієї проблеми безпосередньо застосовуються при навчанні штучних нейронних мереж.

Розглянемо найпростіший алгоритм оптимізації, який широко використовується в навчанні нейронних мереж. Існують також декілька його модифікацій, які збільшують його ефективність.

Градієнтний спуск, також відомий як крутий спуск, є найпростішим тренувальним алгоритмом. Він вимагає інформації з градієнтного вектора, а, отже, це метод першого порядку.

Позначимо $\mathrm{f}(w_i) = f_i$ і $\nabla \mathrm{f}(w_i) = g_i$. Метод починається в точці w_0 і, доки не буде виконано критерій зупинки, рухається від w_i до w_{i+1} у тренувальному напрямку $d_i = -g_i$. Тому метод градієнтного спуску повторюється за формулою (1.9)

$$w_{i+1} = w_i - g_i\alpha_i, \qquad i = 0,1,\ldots \tag{1.9}$$

Параметр швидкості навчання може бути встановлене на фіксоване значення або знайдено одномірною оптимізацією впродовж навчального процесу на кожному кроці. Взагалі переважне значення має оптимальне значення для швидкості тренування, отриманого шляхом мінімізації лінії на кожному наступному етапі. Проте існує багато програмних засобів, які використовують фіксоване значення для швидкості навчання.

Алгоритм навчання градієнтного спуску має серйозний недолік, що вимагає багато ітерацій для функцій, які мають довгі, вузькі “долинні” структури. Справді, градієнт униз - це напрям, в якому функція втрат зменшується найшвидше, але це не обов’язково забезпечує найшвидшу конвергенцію. Рисунок 1.22 ілюструє цю проблему.

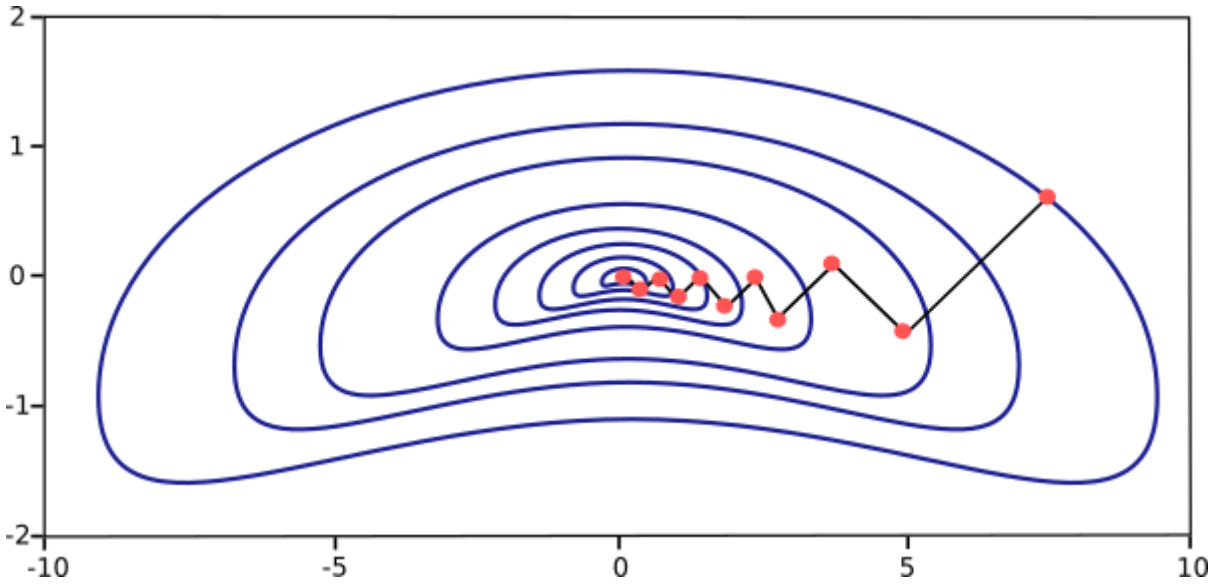

Рисунок 1.22 - Пошук мінімуму за допомогою градієнтного спуску [15]

Градієнтний спуск є рекомендованим алгоритмом, коли у нас дуже великі нейронні мережі, з багатьма тисячами параметрів. Причиною є те, що цей метод зберігає лише градієнтний вектор (розмір n) і не зберігає матрицю Гессіана (розмір n^2).

1.2.5 Регуляризація

Регуляризація є одним з ключових елементів машинного навчання, зокрема глибокого навчання, що дозволяє добре узагальнювати невидимі дані навіть під час тренувань на кінцевому тренувальному наборі або з недосконалою методикою оптимізації [16]. У традиційному сенсі оптимізації, а також у старших нейронних мережах термін “регуляризація” закріплений виключно за величину покарання функції втрати [17]. Останнім часом термін набув більш широкого значення: його легко визначити як “будь-яку модифікацію, яку ми робимо для алгоритму навчання, який призначений для зменшення його тестової помилки, а не його помилки навчання”. Це визначення дещо обмежує і наше робоче визначення регуляризації, оскільки багато методів, які розглядаються як регуляризація, зменшують помилку тренувань. Регуляризація - це будь-яка додаткова техніка, яка спрямована на кращу генералізацію моделі, тобто для отримання кращих результатів на тестовому наборі.

Це може включати різні властивості функції втрат, алгоритм оптимізації втрат або інші методи.

1.2.5.1 Регуляризація через дані

Це робиться шляхом застосування деяких перетворень до навчального набору D в результаті чого створюється новий набір D_R. Деякі перетворення виконують функції вилучення або попереднього обчислення, модифікуючи простір функцій або розподіл даних для деяких реплікацій, спрощуючи завдання навчання. Інші методи дозволяють генерувати нові зразки для створення більшого, можливо, нескінченного, розширеного набору даних. Ці два принципи є дещо незалежними і можуть бути об’єднані. Метою регуляризації через дані є або один з них, або інший, або обидва.

1.2.5.2 Регуляризація через архітектуру мережі

Припущення про відображення: архітектура мережі f може бути вибрана для певних властивостей або відповідати певним припущенням, щоб мати регуляризуючий ефект.

Відображення f_w з входом-виходом має мати певні властивості, щоб відповідати величині даних P. Хоча це може бути складним для забезпечення точних властивостей ідеального відображення, їх можна наблизити спрощеними припущеннями про відображення. Ці властивості та припущення можуть бути застосовані до монтажу моделей твердим чи м’яким способом. Це обмежує простір пошуку моделей і дозволяє знаходити кращі рішення. Прикладом може служити рішення про кількість шарів та одиниць, що дозволяє відображення не бути надто простим або занадто складним (таким чином, уникаючи підміни та перестановки). Іншим прикладом є певні інваріанти відображення, такі як локалізація та еквівалентність зсуву витягання об’єктів, твердо виконані в згорткових шарах. Вибір архітектури f, з одного боку, забезпечує певні властивості відображення; Крім того, при взаємодії між f та алгоритмом оптимізації певні конфігурації маси більш доступні

для оптимізації, ніж інші, і надалі обмежує ймовірність пошуку простору м’яким способом. Доповнюючим способом нав’язування певних припущень про відображення є терміни регуляризації, а також інваріанти, що містяться в (доповненому) наборі даних.

Розділення ваг: повторне використання певного тренованого параметра в кількох частинах цієї мережі називається поділом ваги. Це зазвичай робить модель менш складною, ніж використання параметрів, що піддаються навчанню. Прикладом є згорткові мережі. Тут розподіл ваги не просто зменшує кількість ваг, які потрібно вивчати; він також кодує попередні знання про еквівалентність зсуву і еквівалент місцевості вилучення об’єктів. Інший приклад - це розподіл ваги в автоенкодерах.

Функції активації: вибір правильної функції активації дуже важливий. Наприклад, використання ректифікованих лінійних одиниць (ReLUs) покращило продуктивність багатьох глибинних архітектур як у термінах тренувань, так і в точності. Успіх ReLU може бути пов’язаний з тим, що вони допомагають уникнути проблем зникаючого градієнта, а також тим, що вони забезпечують більш виразні сімейства відображень (класична сигмоїдна нелінійність може бути апроксимована з лише двома ReLU, але приймає нескінченну кількість сигмоїдних одиниць для наближення до ReLU), і їх афінна екстраполяція до невідомих областей простору даних, здається, забезпечує краще узагальнення на практиці, ніж “екстраполяція сигмовидних одиниць”. Деякі функції активації були спеціально розроблені для регуляризації. Dropout Maxout одиниці [18] дають змогу точніше наблизити геометричне середнє прогнозування ансамблю моделі під час тесту. Стохастичний пул, з іншого боку, є шумною версією max-pooling. Це дозволяє моделювати розподіли активацій замість отримання максимального.

Шумні моделі: стохастичне об’єднання було одним з прикладів стохастичного узагальнення детерміністичної моделі. Деякі моделі стохастичні, вводячи випадковий шум у різні частини моделі. Найбільш часто використовувана шумна модель – dropout.

Багатофункціональне навчання: спеціальним типом регуляризації є багатозадачне навчання. Це може поєднуватися з напівавтоматичним навчанням для

використання нерозмічених даних за допоміжним завданням. Подібна концепція обміну знаннями між задачами також використовується в мета-навчанні, де послідовно вивчають декілька завдань з одного і того ж домену, використовуючи раніше отримані знання як упередження для нових завдань; і перенесення навчання (transfer learning), де знання з одного домену переносяться в інший домен.

Вибір моделі: найкраще серед декількох навчених моделей (наприклад, з різними архітектурами) можна вибрати, оцінюючи результат на валідаціному наборі. Слід зазначити, що це стосується вибору найкращого поєднання всіх методів, а не тільки архітектури; і що набір перевірок, який використовується для вибору моделі в “зовнішньому циклі”, має відрізнятися від встановленого валідаційного набору, наприклад, для ранньої зупинки, і відрізняється від тестового набору. Проте існують також моделі вибору моделі, які спеціально призначені для вибору кількості одиниць у певній архітектурі мережі, наприклад. використовуючи розробок мережі та обрізку мережі, або додатково не вимагають встановлення валідації, наприклад Критерій інформаційної мережі для порівняння моделей на основі помилки тренування та другої похідної функції втрат.

1.2.5.3 Регуляризація через функцію помилки

В ідеалі функція E помилки відображає відповідне поняття якості, а в деяких випадках - і припущення щодо розподілу даних. Типові приклади - середня квадратна помилка або крос-ентропія. Функція помилки E також може мати регуляризаційний ефект. Прикладом може служити оптимізація коефіцієнта Діса, який є жорстким для дисбалансу класів. Більш того, загальна форма функції втрат може бути різною, ніж рівняння. Наприклад, в деяких функціях втрат, які є жорсткими для класового дисбалансу, сума беруться по парних комбінаціях $D \times D$ з навчальних зразків, а не за навчальними зразками. Але такі альтернативи для рівняння досить рідко, і застосовуються подібні принципи. Якщо додаткові завдання додані для регуляризуючого ефекту (багатозадачне навчання), то цілі t змінюються, що складається з декількох завдань, картографічний f_w змінений для створення

відповідного виходу *y*, а *E* змінюється для обліку для модифіковані *t* та *y*. Крім того, існують терміни регуляризації, які залежать від *∂E / ∂x*. Вони залежать від *t* і, таким чином, у нашому визначенні розглядаються як частина *E*, а не *R*.

1.2.5.4 Регуляризація через регуляризаційний параметр

Регуляризацію можна досягти, додавши регуляризатор *R* в функцію втрат. На відміну від функції помилки *E* (яка виражає узгодженість результатів з цілями) термін регуляризації не залежить від цілей. Замість цього він використовується для кодування інших властивостей потрібної моделі, щоб забезпечити індуктивне зміщення (тобто припущення про відображення, відмінне від узгодженості результатів з цілями). Таким чином, значення *R* можна обчислити для незаміченого тестового зразка, тоді як значення *E* не може.

Незалежність *R* від *t* має важливе значення: вона дозволяє додатково використовувати немарковані зразки (напівавтоматичне навчання), щоб поліпшити навчену модель на основі її відповідності деяким бажаним властивостям. Наприклад, напівавтоматичне навчання з сходовими мережами (ladder networks) забезпечує «багатозадачне» навчання. Нерозмічені зразки є надзвичайно корисними, коли мітки зразків є дефіцитними. Інсують Байєсівські методи з поєднанням помічених і немаркованих даних в напівавтоматичному навчанні.

1.2.5.5 Регуляризація через оптимізацію

Методи ініціалізації та “теплого старту”: ці методи впливають на початковий вибір моделі ваги. В даний час найбільш часто використовуваним методом є відбір початкових ваг з ретельно налаштованого розподілу. Існує кілька стратегій, що базуються на виборі архітектури, спрямованих на збереження дисперсії активацій у всіх шарах навколо 1, запобігаючи тим самим зникаючим або вибухаючим активаціям (та градієнтам) у більш глибоких шарах.

Іншим (додатковим) варіантом є попереднє тренування (pre-training) з різними даними, з іншою метою або з частково іншою архітектурою. Це може переосмислити алгоритм навчання до гарного рішення перед тим, як відрегулювати від фактичної мети. Попередня підготовка моделі до іншого завдання в одному домені може призвести до вивчення корисних функцій, що полегшить основне завдання. Проте попередньо підготовлені моделі також часто використовуються як лінивий підхід до проблем, коли варто було б спробувати навчатися з нуля або використовувати ретельну адаптацію домену, передачу навчання або багатозадачні методи навчання. З іншого боку, попередня підготовка або аналогічні методи можуть бути корисною частиною таких методів.

Методи оновлення: цей клас методів впливає на індивідуальні оновлення ваги. Є дві додаткові підгрупи: правила оновлення змінюють форму формули оновлення; ваги та градієнтні фільтри - це методи, які впливають на значення градієнта або ваг, які використовуються в формулі оновлень, наприклад. шляхом введення шуму в градієнт.

Знову ж таки, не зовсім зрозуміло, який з методів тільки прискорює оптимізацію і який фактично допомагає узагальненню. Деякі методи, такі як AdaGrad або Adam, навіть втрачають регуляризаційні можливості SGD.

Методи припинення: існує безліч можливих критеріїв зупинки та вибір правильного моменту для зупинення процедури оптимізації може покращити узагальнення шляхом зменшення помилки, викликаної невідповідністю між мінімізаторами очікуваного та емпіричного ризику: мережа спочатку вивчає загальні поняття, що працюють для всіх зразків від розподілу істинності землі P перед встановленням конкретного зразка D та його шуму.

Найуспішніші та найпопулярніші способи завершення поставили частину помічених даних у бік набору перевірок і використовують її для оцінки продуктивності (помилка перевірки). Найвідомішим прикладом є раннє припинення. У сценаріях, де навчальні дані є дефіцитними, можна вдатися до методів припинення, які не використовують набір перевірок. Найпростіший спосіб - встановити кількість проходів через набір тренувань.

1.3 Спайкові нейронні мережі

Багато хто чув про досягнення нинішніх штучних нейронних мережах другого покоління, що використовуються для машинного навчання. Вони, як правило, повністю з’єднані, приймають постійні значення та виводять безперервні значення [19]. Хоча вони дозволили нам досягти прогресу в досягненні прогресу в багатьох областях, вони не є біологічно точними і насправді не імітують реальні механізми нейронів нашого мозку.

Третє покоління нейронних мереж, імпульсні або спайкові нейронні мережі (spiking neural network, SNN), спрямовані на подолання розриву між неврологією та машинним навчанням, використовуючи біологічно-реалістичні моделі нейронів для проведення обчислень. Спайкові нейронні мережі принципово відрізняються від нейронних мереж, які знають спільнота машинного навчання. SNN працюють за допомогою імпульсів, які є дискретними подіями, які відбуваються в точках часу, а не постійні значення. Виникнення сплеску визначається диференційними рівняннями, що представляють різні біологічні процеси, найважливішим з яких є мембранний потенціал нейрона. По суті, як тільки нейрон досягне певного потенціалу, він спрацьовує, і потенціал цього нейрона скидається. Окрім того, SNN часто бувають рідко зв’язними та користуються перевагами спеціалізованих мережевих топологій.

На перший погляд, це може здатися кроком назад. Ми перейшли від безперервних виходів до бінарних, і ці ряди імпульсів не дуже зрозумілі. Проте імпульси надають нам розширену здатність обробляти просторово-часові дані або, іншими словами, сенсорні дані реального світу. Просторовий аспект стосується того факту, що нейрони зв’язуються тільки з локальними для них нейронами, тому вони по суті обробляють шматки вхідного сигналу окремо (аналогічно тому, як згорткова нейронна мережа буде використовувати фільтр). Часовий аспект стосується того факту, що імпульси генеруються з часом, та те що ми втрачаємо в бінарному кодуванні, ми отримуємо через часові властивості спайків. Це дозволяє нам природно обробляти часові дані без додаткової складності, які додають рекурентні нейронні

мережі. Фактично, доведено, що спайкові нейрони принципово більш потужні обчислювальні одиниці, ніж традиційні штучні нейрони.

Враховуючи те, що ці SNN є більш потужними, теоретично, ніж мережі 2-го покоління, природно дивуватися, чому ми не бачимо їх широкого використання. Головне питання, яке наразі полягає в практичному використанні SNN - це навчання. Незважаючи на те, що у нас є безконтрольні біологічні методи навчання, не існує жодних відомих ефективних контрольованих методів навчання для SNN, які забезпечують більш високу продуктивність, ніж мережі 2-го покоління. Оскільки спайки не диференційовані, ми не можемо тренувати SNN з використанням градієнтного спуску, не втрачаючи точної часової інформації в імпульсах. Тому, щоб правильно використовувати SNN для реальних завдань, нам потрібно було б розробити ефективний контрольований метод навчання. Це дуже складне завдання, оскільки це буде означати визначення алгоритму навчання людського мозку, беручи до уваги біологічний реалізм у цих мережах.

Ще одне питання, що ми набагато ближче до вирішення, полягає в тому, що моделювання SNN на звичайному апаратному забезпеченні потребує дуже складних обчислень, оскільки вимагає симуляцію диференціальних рівнянь.

Тому майбутнє SNN залишається незрозумілим. З одного боку, вони є природним наступником наших нинішніх нейронних мереж, але, з іншого боку, вони далекі від практичних інструментів для більшості завдань. Існує декілька сучасних застосунків SNN для обробки зображень та аудіо в режимі реального часу, але література з практичних застосувань залишається незначною. Більшість робіт зі SNN є або теоретичними, або показують продуктивність таку ж, як проста повнозв’язна мережа 2-го покоління.

1.3.1 Кодування

Метою штучних імпульсних нейронних мереж є проведення нейронних обчислень. Це вимагає того, щоб значення було подане до нейронних спайків: значення, що мають відношення до обчислень, повинні бути виражені у формі

імпульсів, за допомогою яких нейрони спілкуються. Проблема полягає в тому, що природа нейронних кодів є невирішеною темою досліджень в галузі нейронаук. Однак, виходячи з того, що відомо з біології, було запропоновано ряд нейронних інформаційних кодів [20].

Бінарне кодування (binary) - це кодування, де все або ніщо: кожен нейрон активний протягом певного проміжку часу, тобто він випускає один або декілька імпульсів у цей часовий інтервал або він мовчить. Це кодування мотивоване спостереженням, що фізіологічні нейрони схильні спрацювати, коли вони отримують вхідний сигнал (сенсорний подразник, такий світло або зовнішні електричні подразники). Ця двійкова абстракція може бути використана на рівні індивідуальних нейронів: нейрони моделюються як двійкові одиниці, які можуть бути ввімкненими або вимкнутими. Ця модель була використана вже в моделях ранніх нейронних мереж, таких як мережі ADALINE. Вона також може використовуватися на рівні інтерпретаційних спайкових послідовностей з сучасних спайкових нейронних мережі, де бінарна інтерпретація вихідних послідовностей використовується в класифікації входів спайкових послідовностей. Таким чином, двійкове кодування було використано в сучасних реалізаціях спайкових нейронних мереж. Взагалі, бінарне кодування є привабливим через його простоту, але воно ігнорує синхронний характер і множинність імпульсів взагалі.

Швидкісне кодування (rate) є іншою абстракцією від синхронного характеру імпульсів, оскільки лише швидкість імпульсів в інтервалі використовується як міра для переданої інформації. Швидкісне кодування мотивується спостереженням, що фізіологічні нейрони мають тенденцію до спрацювання частіше для сильних (сенсорних чи штучних) подразників. Це може знову бути застосоване на рівні одиночного нейрона, або при інтерпретації спайкових послідовностей. У першому випадку нейрони моделюються безпосередньо за швидкістю нейронів, які передають на кожному етапі реальні значення вхідних чисел – “ставки” у вихідні “швидкість”. Поняття швидкісного кодування виходить за межі стандартних штучних сигмоідальнимих нейронів в технічних контекстах і когнітивній науці. Для інтерпретації спайкових послідовностей кодування швидкості (також зване

частотним кодуванням) також використовується для інтерпретації виходів спайкових нейронних мереж.

Кодування затримкою (latency) використовує час імпульсів, але не множинність імпульсів. Інформація кодується як затримка від певної (внутрішньої або зовнішньої) події до першого спрацювання. Це мотивовано спостереженням, що важливі сенсорні подразники, як правило, викликають імпульс раніше в upstream нейронах. Це кодування, наприклад, використовувалося в навчанні без учителя та навчанні з учителем, такими як SpikeProp або Chronotron та інші. Тісно пов’язане це кодування з кодуванням рангового порядку (rank-order), де інформація про стимул кодується в тому порядку, в якому нейрони всередині групи випускають свої перші імпульси.

Повністю часові коди (fully temporal). Всі згадані раніше кодування є особливими випадками повністю часового коду. У повністю часовому коді кодування залежить від точного часу всіх спайків. Вони взяті з доказів у неврології, що час імпульсу може бути надзвичайно точними та відтворюваним. У повністю часовому коді час відносний щодо певної заданої (внутрішньої або зовнішньої) події (наприклад, поява стимулу або імпульсу референтного нейрону).

Передбачаюче імпульсне кодування (predictive spike) - це особливий випадок повного часового кодування. Тут випливає, що імпульсний механізм ефективно забезпечує засіб для здійснення аналого-цифрового та цифро-аналогового перетворення на сомах та синапсах нейрона, відповідно. Приклади включають в себе жадібні рішення, де генерація спайнів в сомах пов’язана з відніманням тимчасового ядра з вхідного струму; в синапсі кожний спайк потім викликає потенціювання у цільовому нейроні так, що сума потенціалів наближає обчислений сигнал до соми. Механізми адаптації можуть змінювати потенціал, що дозволяє спайковому нейрону пристосуватися до змін у динамічному діапазоні входів. Пристосована динаміка спайкових нейронів дозволяє їм робити обчислення за декількома часовими масштабами.

Імовірнісне кодування імпульсів (probabilistic spike) полягає в тому, щоб ефективно робити судження з використанням спайкових нейронів, і це в основному розглядається в контексті обчислювальної нейробіології. Тут є два відомі способи

розробок нейронних розрахунків: де розглядаються спайкові нейрони, які стохастично генерують імпульси у відповідь на заданий вхід та, де спайкові нейрони суттєво детерміновані. В обох випадках обчислюється неявна міра судження, перша в швидкісному коді і останній в прогнозному коді спайка.

1.3.2 Нейрони

Модель нейрона забезпечує формулювання того, як нейрон обробляє вхідні імпульси та як він спрацьовує. Є багато методів [21] для опису цих моделей, а загальні пояснюються нижче.

Модель Ходжкіна-Хаксли (Hodgkin-Huxley, HH) заснована на провідності каналів іонів. Вона моделює нейрон, використовуючи значення ємності та провідності іонних каналів та їх рівноважні потенціали. Диференційні рівняння (1.10) та (1.11) описують динаміку моделі.

$$C\frac{du}{dt} = -g_{N_a}m^3h\left(u - E_{N_a}\right) - g_K n^4(u - E_K) - g_L(u - E_L) + I(t) \qquad (1.10)$$

$$\tau_n\frac{dn}{dt} = [n - n_0(u)], \tau_m\frac{dm}{dt} = -[m - m_0(u)], \tau_h\frac{dh}{dt} = -[h - h_0(u)] \qquad (1.11)$$

За цими рівняннями можна дуже детально моделювати поведінку нейрона, наприклад, раптове збільшення потенціалу при стрільбі, період абсолютної рефрактерності та відносний рефракторний період (рис 1.23). Оскільки ця модель занадто складна, то важко з її допомогою реалізувати великі мережі навіть для симуляції SNN.

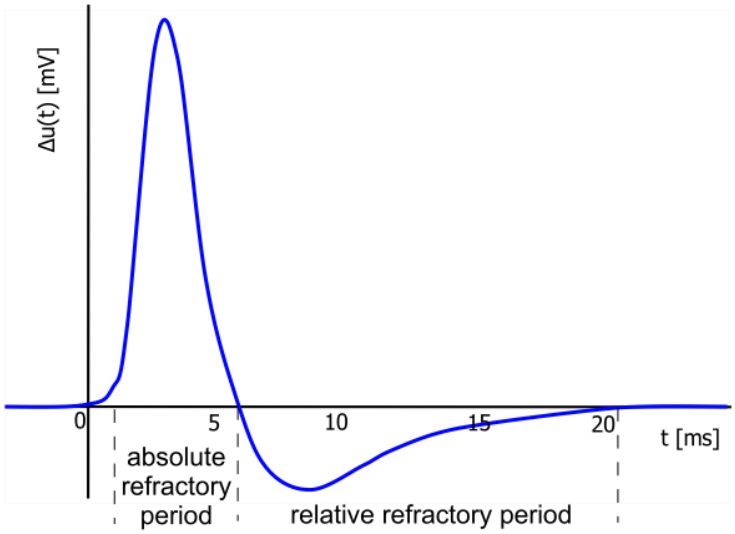

Рисунок 1.23 - Динаміка спайкового нейрона [22]

Integrate-and-Fire (IF) походить від попередньої моделі і має менші обчислювальні витрати. У IF є варіанти, такі як Leaky-Integrate-and-Fire (LIF). LIF ігнорує форму потенціалу дії та звертає увагу на терміни дії спайку. Динаміка нейрона представлена рівнянням (1.12).

$$\tau_m \frac{du}{dt} = u_{rest} - u(t) + RI(t) \tag{1.12}$$

Потенціал дії скидається на “відпочинок” відразу після спрацювання. Абсолютний рефракторний період можна визначити як $u = -u_{abs}$ протягом періоду після стрільби, а потім встановити потенціал дії до u_{rest}.

Модель Quadratic-Integrate-and-Fire (QIF) залежить від квадрата потенціалу дії. На додаток до LIF, модель QIF здатна забезпечувати динамічні властивості імпульсу, такі як затримане спрацювання та залежна від активності границя спрацювання.

Тета-нейронна модель може бути інтерпретована як перетворення моделі QIF, де стан визначається фазою. Як тільки нейрон випускає імпульс, фаза перетинає π (рис. 1.24).

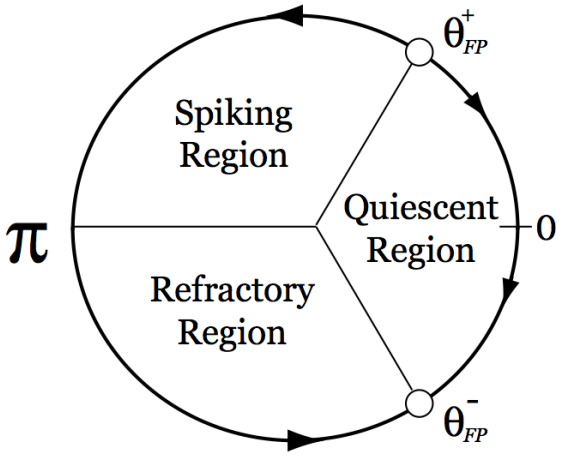

Рисунок 1.24 - Візуалізація динаміки нейрона за допогою кола [22]

Динаміка нейрона описується рівнянням (1.13).

$$\frac{d\theta}{dt} = (1 - \cos(\theta)) + \alpha I(t)(1 + \cos(\theta)) \tag{1.13}$$

Модель Іжикевича забезпечує баланс між обчислювальною вартістю та біологічною реальністю. Дійсно, можна описати багато різних спрацювань нейронів, використовуючи модель Іжикевича.

Spike Response Model (SRM) визначає потенціал дії як інтеграл у минулому з урахуванням періоду рефракторингу. Вона заснована на появі імпульсів. SRM модель здатна імітувати складні розрахунки, навіть якщо вона має надзвичайно простішу структуру, ніж модель HH.

1.3.3 Ваги

Ваги в спайкових нейронних мережах відіграють таку ж роль, як і в звичайних штучних нейронних мережах. Вони з’єднують нейрони і показують на скільки один нейрон сильно впливає на інший. В термінах спайкових (та біологічних) нейронних мережах їх ще називають синапсами. Синаптична пластичність означає встановлення ваги синапсів, що є основою навчання мережі.

1.3.4 Структура

Спайкові нейронні мережі не мають певної структури, нейрони можуть бути хаотично сполучені з іншими нейронами, як це відбувається в біологічних нейронних мережах. Зазвичай при побудові мережі для подальшої симуляції виділяють шари, як і в звичайних штучних нейронних мережах, і вже оперують групою нейронів які зв’язані з нейронами інших шарів.

Спайкові нейронні мережі дозволяють також утворювати рекурентні зв’язки між нейронами. Ось так можна, наприклад, симулювати процес бічного гальмування (lateral inhibition). У нейробіології бічне гальмування - це здатність збудженого нейрона знизити активність сусідів. Бічне гальмування блокує поширення потенціалів дії від збуджених нейронів до сусідніх нейронів у бічному напрямку [23]. Це створює контраст у стимуляції, що дозволяє збільшити сенсорне сприйняття. Він також називається бічним антагонізмом і виникає насамперед у візуальних процесах, але також в тактильній, слухової та навіть нюхової обробці.

На рисунку 1.25 зображено як стимул, який впливає на всі три нейрони, але який впливає на B найсильнішим або першим, може бути загостреним, якщо B посилає бічні сигнали сусідам A і C, щоб вони не спрацьовували, тим самим гальмуючи їх.

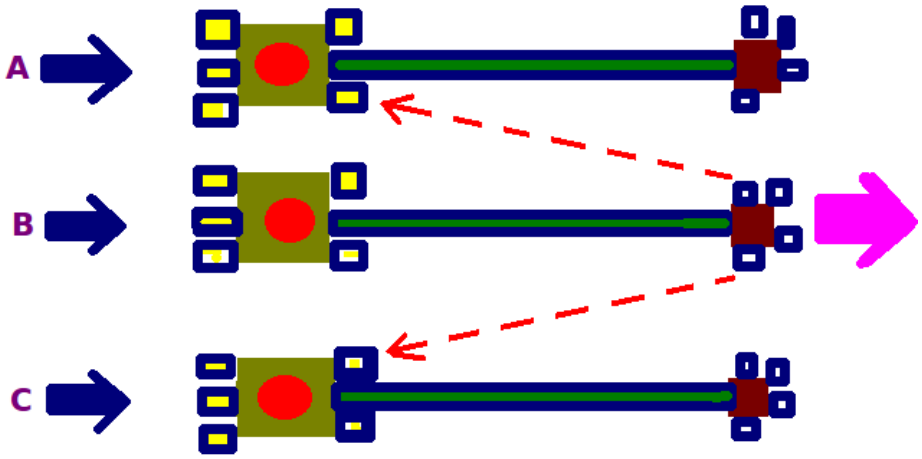

Рисунок 1.25 - Бічне гальмування [23]

1.3.4.1 Резервуарне обчислення

Біологічно натхненні структури, такі як мережі з розріджено з'єднаними нейронами та різними імпульсами можна адаптувати до архітектури спайкових нейронних мереж. Резервуарне обчислення (reservoir computing) визначає сімейство мереж, які обробляють часові патерни з нейронами 3-го покоління (спайковими нейронами). Архітектура мережі резервуарного обчислення складається з трьох основних частин: вхідного шару до резервуара, самого резервуара, який є рекурентною мережею, і вихідного шару (рис 1.26).

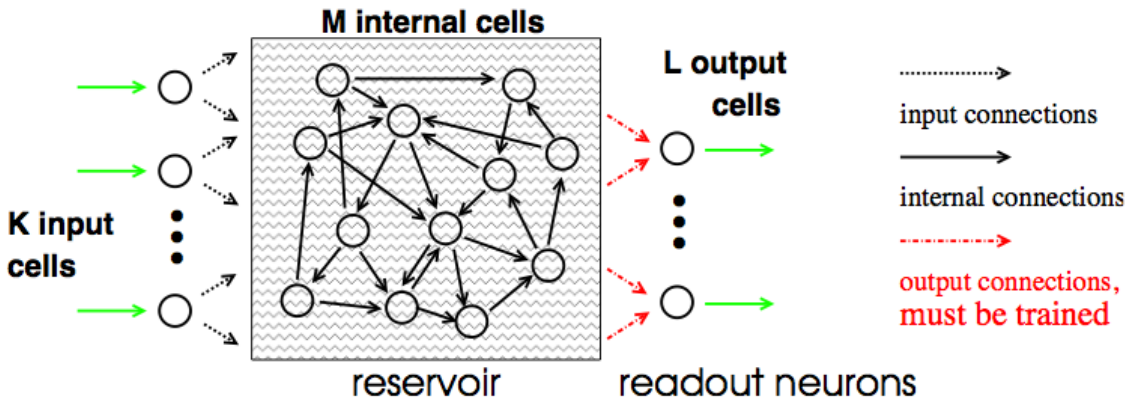

Рисунок 1.26 - Схема резервуарного обчислення [22]

Основною ідеєю обчислення резервуара є вибір відповідної частини з резервуара з нейронами вихідного шару. В деяких мережах резервуар залишається незмінним після встановлення ваг випадковим чином або певним підбором, а тренується лише з'єднання від резервуара до вихідного шару. В інших мережах використовуються алгоритми навчання спайкових нейронних мереж для тренування резервуара, а з'єднання з резервуара до вихідного шару навчаються за допомогою лінійної регресії.

Мережа з відлунням стану (echo state network, ESN) та рідкий скінченний автомат (liquid-state machine, LSM) є базовими моделями обчислень резервуарів, які зручні для експлуатації часових особливостей спайкових нейронів [21].

ESN намагається вивчати часові ряди, використовуючи періодичні мережі. Внутрішній стан резервуара є функцією одночасного введення, колишнього стану та колишнього виходу. Ефективна ESN повинна бути в змозі забувати, що є можливим завдяки коригуванню ваги резервуарів, таким чином, щоб реалізувати поведінку згасаючої пам’яті.

LSM зосереджується на обробці безперервного введення в реальному часі, використовуючи IF нейрони. Резервуар, що також називається “рідкий фільтр”, перетворює вхід у “рідкий стан”, з якого генерується вихід з використанням “безпам’ятно-зчитувальні карти”. Мапа зчитування повинна виробляти стабільний вихід з резервуара. Маючи декілька карт зчитування, можна здійснювати паралельне обчислення в режимі реального часу.

Архітектури резервуарних обчислень зручні для прогнозування часових рядів та розпізнавання часових моделей. Як LSM, так і ESN можуть обробляти часові зміни інформації.

Застосування LSM зазвичай відбувається в біологічно схожих моделях. Оскільки спайкова нейронна мережа має часову динаміку, вона підходить для послідовних даних. Таким чином, LSM з імпульсними нейронами може призвести до задовольняючих моделей розпізнавання мови. Відповідно, архітектури SNN також можуть забезпечувати рішення в комп’ютерному зорі з використанням асинхронності імпульсів.

1.3.5 Навчання

1.3.5.1 Навчання з учителем

Найпростіший підхід до здійснення керованого навчання в розповсюдженні нейронних мереж полягає в наступному: використовувати градієнтний підхід до зниження похибки в мережі [24]. Цей підхід більш складний, аніж традиційне зворотне поширення, оскільки активаційна функція нейрона не є диференційованою, але в [25] було отримано алгоритм, подібний до зворотного розповсюдження, для спайкових мереж з деякими додатковими припущеннями. Їх алгоритм, що

називається SpikeProp, працює так само, як традиційне зворотне поширення, оскільки він обчислює загальну похибку - різницю часу між імпульсами, створеними мережею і бажаними імпульсами, - і призначає локальну помилку для кожного вузла, який використовується для зміни з'єднання ваги пропорційно активності вузла. Проте, як і зворотне поширення, локальна помилка для кожного вузла залежить від ваги з'єднання нижніх нейронів, що робить цей алгоритм біологічно неправдоподібним.

Більш пізніший алгоритм контрольованого навчання спайкрвих мереж - це метод віддаленого контролю Remote Supervised Method (ReSuMe). Правило навчання, яке використовує ReSuMe, біологічно схоже, описане рівняннями (1.14) та (1.15).

$$\Delta w_{i,j} = -\Delta_{wij}{}^{STDP} + \Delta_{wk}{}^{STDP} \tag{1.14}$$

$$\Delta w_{i,j} = \Delta_{wij}{}^{STDP} - \Delta_{wk}{}^{STDP} \tag{1.15}$$

ReSuMe також використовує не Геббінський параметр для прискорення навчання, але навіть без цієї змінної, доводиться, що ReSuMe може навчитися модифікувати синаптичні ваги таким чином, щоб вихідний нейрон спрацював у потрібний час, отримавши ті ж самі вхідні імпульси.

Модель навколоносової кори, яка навчається за допомогою правила, схожого на зворотного розповсюдження називається FreqProp. У цій моделі ваги з прихованого шару до вихідного шару фіксуються, згідно зі спостереженням, що в навколоносової корі нейрони, що спрацьовують залежно від новизни подразника, обчислюють тривіальну схему підсумовування голосів, а отже, і ваги з'єднання з прихований до вихідного шару фіксуються і можуть бути проігноровані при розрахунку правила навчання, що змінюють синапси вхідного-прихованого шарів. Це правило навчання виражене у рівнянні (1.16).

$$\Delta w_{i,j} = \frac{1}{N} \delta \alpha_i \alpha_j \tag{1.16}$$

Назва FreqProp випливає з того, що частота спрацювання нейронів вихідного шару модулює кількість синаптичних змін у попередніх шарах. Залежність від рівності мас синапсів прихованого-вихідного шарів визначає біологічну правдоподібність правила, і це випливає з спостережень навколоносової кори, що навряд чи буде вірно для інших кортикальних ділянок.

1.3.5.2 Навчання без учителя

Правило Artola, Bröcher, Singer (ABS) моделює результати, виявлені в Artola, Bröcher і Singer у візуальній корі щура. Вони встановили, що напрямок синаптичної зміни ваги залежить від швидкості постсинаптичного збільшення. Найпростіше правило навчання, яке охопило їх якісні результати описане рівнянням (1.17).

$$\Delta w_{i,j} = \alpha_i \sigma(\alpha_j), \qquad \text{де } \sigma(\alpha_j) = \begin{cases} \Delta^+ \text{ якщо } \alpha_j \geq \theta^+ \\ \Delta^- \text{ якщо } \theta^- < \alpha_j < \theta^+ \\ 0 \text{ в інших випадках} \end{cases} \tag{1.17}$$

Правда ABS, не має широкого застосування в моделюванні, і ми не можемо знайти досліджень, які аналізують його функціональне значення в розширеній нейронній моделі [24]. Це може бути пов'язано з тим, що він має небагато функціональних переваг або тому, що він дуже схожий з правилом BCM, яке було запропоноване раніше, тільки без гарантованої стабільності.

Також було запропоновано правило навчання, яке моделює результати біологічних досліджень LTP / LTD. Воно схоже на правило ABS, оскільки воно фільтрує постсинаптичну активність з кривою, яка має початковий негативний компонент, а потім позитивний компонент; однак, на відміну від ABS, існує лише один поріг, який ми називаємо θ. Потрібно лише один поріг, тому що з будь-якою постсинаптичною активністю буде деяка зміна ваги, позитивна або негативна (поки не $a_j = \theta$); це суперечить висновкам деяких статей, але збігається з іншими експериментальними даними та спрощує модель. Подібно до більш складної версії

правила ABS, поріг змінюється, в цьому випадку як функція очікуваної вартості постсинаптичної активності. Іншими словами, цей поріг містить деяку інформацію про історію діяльності клітини, а відхилення від історичних патернів спрацювання призводять до синаптичних змін ваги.

Перші правила spike-timing-dependent plasticity (STDP) були засновані на взаємодії між парами спайків: попереднє спарювання викликає потенціювання, а наступне спрацювання викликає депресію. Це можна виразити наступними двома рівняннями (1.18) і (1.19).

$$\Delta w_{i,j}(t^{pre}) = A^{-} \exp\left(\frac{t^{post_l} - t^{pre}}{\tau^{-}}\right) \tag{1.18}$$

$$\Delta w_{i,j}(t^{post}) = A^{+} \exp\left(\frac{t^{pre_l} - t^{post}}{\tau^{+}}\right) \tag{1.19}$$

Це правило навчання лише враховує найбільш недавній пресинаптичний і постсинаптичний імпульс; однак, ми знаємо, що різні частоти діяльності впливають на пластичність. Першим кроком до моделювання цього є включення більш ніж одного спайку, використовуючи сліди діяльності для попередньої та постсинаптичної дії.

STDP застосовується для багатьох різних моделей, і до нього відносяться багато різних функціональних наслідків, що вказує на широкий спектр можливих функцій вартості, які можуть бути мінімізовані.

1.3.5.3 Навчання з підкріпленням

Незважаючи на те, що навчання з підкріпленням - це більш складна проблема, ніж контрольоване навчання, кількість літератури для вивчення навчання з підкріпленням в експериментальній і теоретичній нейронауках є величезною і швидко розвивається [24].

Витоки навчання з підкріпленням знаходяться в класичних експериментах з підготовки Павлова, які показали, що тварини можуть навчатись певним чином реагувати на довільні стимули. Класичний експеримент, виконаний Павловим, поєднував слуховий тон з доставкою їжі до голодних собак. Після повторення тону з доставкою їжі у собак починалось виділення слини після тону, але до того, як їжа була доставлена. Ми називаємо аудиторний тон умовним стимулом (УС) і доставкою їжі безумовним подразником (БП). У БП, як правило, є якесь внутрішнє позитивне чи негативне значення для тварини, тоді як УС є довільним; класичний умовний рефлекс може бути використаний для передачі деяких аспектів БП в УС. Собаки, що у яких виділялась слина після тону, але до того, як було доставлено їжу, вказують на те, що собаки відреагували на тон, спрогнозувавши доставку їжі, що викликало вегетативну реакцію. У класичному умовному рефлексі тварина не повинна виконувати дії, щоб забезпечити доставку БП, тому вона навчається поєднувати два стимули; це іноді називають навчання стимул-стимул асоціаціям.

Більшість моделей спайкових нейронних мереж навчання з підкріпленням використовують правило, засноване на модульованій винагороді правила STDP, щоб сильніше зміцнити ті нейрони, які активні в певному бажаному стані.

1.4 Висновки за розділом

Нейронні мережі є одними із найпотужніших методів машинного навчання, які широко використовуються незалежно від типу навчання.

В даному розділі було розглянуто основні типи машинного навчання та їх особливості. Було проаналізовано основні компоненти штучних та спайкових нейронних мереж. Незважаючи на велику відмінність даних мереж, при їх побудові та навчанні зазвичай розглядаються ті ж самі компоненти: тип нейрона, ініціалізація ваг, структура та правило навчання. Зазвичай спайкові нейронні мережі не використовують для практичного машинного навчання, адже симуляція потребує багато ресурсів, а показники в них не кращі за класичні штучні нейронні мережі.

Спайкові нейронні мережі більш цікаві з точки зору розуміння як біологічні нейронні мережі навчаються та працюють.

2 АНАЛІЗ МЕТОДІВ ПОПЕРЕДНЬОГО ТРЕНУВАННЯ

Ідея попереднього тренування досить проста: до того як тренувати модель певним чином на певних даних ми її грамотно ініціалізуємо, теж натренувавши. В такий спосіб можна отримати більш точну модель або модель, яка швидше навчається.

Для деяких задач це є критичним, наприклад, де в наявності немає достатньої кількості розмічених даних (напівавтоматичне навчання).

Також, натреновані моделі можна використовувати в схожих задачах (transfer learning) де їх потім підлаштовують повністю або лише останні шари. На практиці дуже мало людей навчають цілу згорткову нейронну мережу з нуля (з випадковою ініціалізацією), тому що досить рідко мають достатню кількість даних. Натомість загальноприйнятним способом є попереднє тренування ConvNet на дуже великому наборі даних (наприклад, ImageNet, що містить 1,2 мільйона зображень з 1000 категоріями), а потім використовувати ConvNet як ініціалізацію або закріплений (фіксований) екстрактор ознак для поставленої задачі. Виділяють три основні сценарії навчання через перенесення [26]:

- ConvNet як фіксований екстрактор ознак. Береться згорткова мережа, попередньо навчена на наборі ImageNet, видаляється останній повністю підключений шар (виходи цього шару - класи для іншого завдання, такого як ImageNet), а потім сприймати решту згорткової мережі як фіксований екстрактор ознак для нового набору даних. Ці ознаки також називають CNN кодами;

- Підлаштування ConvNet. Друга стратегія полягає в тому, щоб не тільки замінити та підлаштувати класифікатор на верхньому шарі ConvNet на новому наборі даних, але також оптимізувати ваги попередньо навченої мережі, продовжуючи оптимізацію. Можна точно налаштувати всі шари ConvNet, або можна зберегти деякі попередні шари (через проблеми з перенавчанням) і лише точно налаштувати частину вищого рівня мережі. Це мотивовано спостереженням, що попередні ознаки ConvNet містять більше загальних ознак (наприклад, детектори краю або кольорові детектори кольорів), які повинні бути корисними для багатьох завдань, але згодом шари

ConvNet стають поступово більш специфічними для деталей класів що міститься в оригінальному наборі даних. У випадку з прикладом ImageNet, що містить багато собачих порід, значна частина представницької потужності ConvNet може бути присвячена ознакам, специфічним для диференціації між породами собак;

- Попередньо натреновані моделі. Оскільки сучасні ConvNets забирають 2-3 тижні для навчання на кількох графічних процесорах на ImageNet, загальноприйнятим є те, що люди діляться своїми контрольними точками навчання ConvNet на користь тих, хто може використовувати мережі для підлаштування. Наприклад, у бібліотеці Caffe є модельний зоопарк, де люди діляться своїми вагами мережі.

2.1 Попереднє тренування з учителем

Іноді безпосереднє тренування моделі до вирішення конкретної задачі може бути надто амбіційним, якщо модель складна і важко її оптимізувати, або якщо завдання складне [4]. Іноді більш ефективно підготувати просту модель для вирішення завдання, а потім зробити модель більш складною. Також може бути більш ефективним навчання моделі для вирішення більш простих завдань, а потім перейти до виконання остаточного завдання. Ці стратегії, що передбачають підготовку простих моделей до простих завдань перед тим, як зіткнутися із завданням навчання потрібної моделі для виконання бажаної задачі, загалом називають переднавчанням або попереднім тренуванням.

Жадібні алгоритми розбивають проблему на багато компонентів, а потім вирішують для оптимальної версії кожного компонента окремо. На жаль, об'єднання індивідуально оптимальних компонентів не гарантує отримання оптимального повного рішення. Проте жадібні алгоритми можуть бути обчислювальними значно дешевими, ніж алгоритми, які вирішують найкраще спільне рішення, а якість жадібного рішення часто є прийнятним, якщо не оптимальним. Жадібні алгоритми можуть також супроводжуватися етапом точного налаштування, в якому алгоритм спільного оптимізації шукає оптимальне рішення для повної проблеми. Ініціалізація

спільного алгоритму оптимізації з жадібним рішенням може значно прискорити його та покращити якість рішення, яке він знаходить.

Підготовка, і особливо жадібна підготовка – широкомасштабні алгоритми в глибокому вивченні. Нижче описано ті алгоритми попереднього тренування, які розбивають задачі навчання з учителем на інші простіші задачі навчання з учителем. Цей підхід називається жадібним попереднім тренуванням з учителем.

У оригінальній версії жадібної попередньої підготовки з учителем кожен етап складається з завдання контрольного навчання, що включає лише підмножину шарів у кінцевій нейронній мережі. Замість попереднього тренування одного шару за раз, тренують глибоку CNN мережу (одинадцять шарів), а потім використовують перші чотири та три останніх шари з цієї мережі для ініціалізації ще глибших мереж (з дев'ятнадцятьма шарами ваг). Середні шари нової, дуже глибокої мережі ініціалізуються випадковим чином. Потім нова мережа навчається спільно. Інший варіант полягає у використанні результатів раніше навчених нейронних мереж, а також незмінених даних, як вхідних даних для кожної додаткової стадії.

Чому жадібний алгоритм навчання з учителем допомагає? Гіпотеза, спочатку полягає в тому, що він допомагає забезпечити краще управління проміжними рівнями глибокої ієрархії. Загалом, попередня підготовка може допомогти як з точки зору оптимізації, так і з точки зору узагальнення.

Підхід, пов'язаний з контролем за попередньою підготовкою, розширює ідею в контексті навчання через передачу (transfer learning). Попередньо натренуємо глибоку згорткову мережу з 8 шарів ваг на сукупність завдань (підмножина з 1000 категорій об'єктів ImageNet), а потім ініціалізуємо мережу з однаковими розмірами з першими k-шарами першої мережі. Всі шари другої мережі (з верхніми шарами ініціалізованими випадковим чином), потім спільно навчені для виконання іншого набору завдань (інша підмножина тисячі категорій об'єктів ImageNet), з меншою кількістю навчальних прикладів, ніж для першого набору завдань.

Ще однією схожою роботою є підхід FitNets. Цей підхід починається з тренування мережі, яка має достатньо низьку глибину та досить широку ширину (кількість одиниць на шар), щоб бути легко натренованою. Ця мережа потім стає

вчителем для другої мережі, яка називається учнем. Мережа-учень набагато глибша і тонша (від одинадцяти до дев'ятнадцяти шарів), і в звичайних умовах важко буде тренуватися за допомогою стохастичного градієнтного спуску. Навчання мережі-учня полегшується завдяки навчанні мережі-учня не тільки для прогнозування результатів вихідного завдання, але й для прогнозування значення середнього шару мережі-вчителя. Це додаткове завдання надає набір порад про те, як слід використовувати приховані шари, і може спростити проблему оптимізації. Вводяться додаткові параметри для регресу середнього шару 5-рівневої мережі-вчителя із середнього шару глибшої мережі-студента. Проте, замість прогнозування фінального класу, метою є прогнозування середнього прихованого шару мережі-вчителя. Таким чином, нижчі шари мережі-учня мають дві цілі: допомогти виходам мережі-учня виконати завдання, а також передбачити проміжний рівень мережі-вчителя. Незважаючи на те, що тонка і глибока мережа, як видається, важче тренується, ніж широка і неглибока мережа, тонка і глибока мережа може краще узагальнюватися і, звичайно, має меншу обчислювальну вартість, якщо вона досить тонка, щоб мати набагато менше параметрів. Без рекомендацій на прихований шар, мережа-учень працює дуже погано в експериментах, як на навчанні, так і на тестовому наборі. Поради щодо середніх шарів, таким чином, можуть бути одним з інструментів, що допомагають навчати нейронні мережі, які інакше здаються важкими для навчання, але інші способи оптимізації або зміни в архітектурі також можуть вирішити цю проблему.

2.2 Попереднє тренування без учителя

Нейронні мережі мають тисячі, часто мільйони параметрів. Вони приймають сотні ознак і класифікують тисячі класів. Функції часто не можуть розглядатися незалежно, але їх потрібно враховувати в цілому. Більшість параметрів також не є незалежними. І все-таки ми використовуємо лише близько десяти тисяч до мільйонів точок даних для оптимізації мільйонів параметрів у мережі. Ми знаємо, що більше міток даних призводять до кращих результатів, але маркування коштує дорого. Однак

одержання більшої кількості даних є порівняно дешевим. Навчання без учителя відіграло ключову історичну роль у відродженні глибоких нейронних мереж, що дозволило дослідникам вперше навчити глибоку контрольовану мережу, не вимагаючи архітектурних спеціалізацій, таких як згортка або рекурентність. Отже, ми хочемо використовувати нерозмічені дані для вивчення хороших ознак [27]. Повний процес підпорядкований і працює таким чином:

- Навчання без учителя: навчити нейронну мережу з нерозміченими даними;

- Модифікація мережі: змінити щось у мережі. Часто вихідний шар регулюється;

- Навчання з учителем: навчити нейронну мережу з позначеними даними.

Жадібне пошарове попереднє тренуванням без учителя - процедура є канонічним прикладом того, як одержуване представлення одного завдання (навчання без учителя, намагаючись відобразити форму розподілу вхідних даних) іноді може бути корисним для іншого завдання (кероване навчання з тим самим вхідним доменом).

Жадібне пошарове навчання без учителя спирається на алгоритм вивчення одношарового представлення, такого як RBM, одношаровий автоенкодер, модель розрідженого кодування або інша модель, яка вивчає приховані представлення [4]. Кожен шар попередньо навчений, використовуючи безконтрольне навчання, виводячи результат попереднього шару і видавши нове представлення, чий розподіл (або його відношення до інших змінних, таких як категорії, для прогнозування), сподіваючись буде простішим.

Жадібне пошарове тренування, засноване на безконтрольному критерії, вже давно був використаний для того, щоб обійти труднощі спільної підготовки шарів глибокої нейронної мережі для контрольного завдання. Відродження глибокого навчання 2006 року розпочалося з відкриття того, що цю жадібну процедуру навчання можна використовувати для того, щоб знайти гарну ініціалізацію для спільної процедури навчання на всіх шарах, і що цей підхід може бути використаний для успішного навчання навіть повністю зв'язаних архітектур. До цього відкриття тільки згорткові глибинні мережі чи мережі, чия глибина була наслідком рекурентності,

вважалися доцільними для тренувань. Сьогодні ми знаємо, що жадібне пошарове попереднє тренування не вимагається для навчання повністю зв'язаних глибоких архітектур, але підхід до попереднього безконтрольного тренування був першим способом досягнення успіху.

Жадібне пошарове попереднє тренування називається жадібним, оскільки це жадібний алгоритм, що означає, що він оптимізує кожен шматочок рішення незалежно, один шматок за раз, а не спільно оптимізує всі шматки. Це називається пошаровим, оскільки ці незалежні частини є шарами мережі. Зокрема, жадібне пошарове попереднє тренування відбувається одночасно з одним шаром, тренуючи k-й шар, залишаючи попередні шари фіксованими. Зокрема, нижні шари (які навчаються спочатку) не адаптуються після введення верхніх шарів. Воно називається безконтрольним, оскільки кожен шар навчається за алгоритмом навчання без учителя. Проте його також називають попереднім тренуванням, оскільки воно повинно бути лише першим кроком до спільного алгоритму тренування, який застосовується для точного налаштування всіх шарів. У контексті завдання контрольованого навчального його можна розглядати як регуляризатор (в деяких експериментах, попередня підготовка зменшує помилку тесту, не зменшуючи похибки навчання) та форму ініціалізації параметрів.

Загальноприйнято використовувати слово “попереднє тренування” для позначення не лише самої стадії попередньої підготовки, але й всього протоколу з двох фаз, що поєднує в собі фазу попередньої підготовки та контрольовану фазу навчання. Фаза навчання з учителем може включати в себе підготовку простого класифікатора на основі ознак, вивчених на етапі попередньої підготовки, або це може включати контрольовану тонке налаштування всієї мережі, яка вивчається на етапі попередньої підготовки. Незалежно від того, який тип алгоритму навчання без нагляду або який тип моделі використовується, у переважній більшості випадків загальна схема навчання майже однакова. Незважаючи на те, що вибір алгоритму навчання без учителя, очевидно, вплине на деталі, більшість застосувань навчання без нагляду з попереднім тренуванням дотримуються цього базового протоколу.

Жадібне пошарове безконтрольне попереднє тренування також може використовуватися як ініціалізація для інших безконтрольних алгоритмів навчання, таких як глибокі автокоенкодери та імовірнісні моделі з багатьма рівнями латентних змінних. Такі моделі включають в себе deep belief networks та глибокі машини Больцмана.

За багатьма завданнями, жадібний пошарове безконтрольне попереднє тренування може призвести до істотного поліпшення тестової помилки для задач класифікації. В багатьох інших завданнях, однак, неконтрольоване попереднє тренування не дає переваг або навіть заподіює значну шкоду. Раніше було вивчено вплив попередньої підготовки на моделі машинного навчання для прогнозування хімічної активності та встановлено, що в середньому попереднє тренування було трохи шкідливим, але для багатьох завдань було значно корисним. Оскільки попереднє навчання без учителя іноді є корисним, але часто шкідливим, важливо зрозуміти, коли і чому воно працює, щоб визначити, чи можна його застосувати до конкретного завдання.

Безконтрольне попереднє навчання поєднує в собі дві різні ідеї. По-перше, воно використовує ідею, що вибір початкових параметрів для глибокої нейронної мережі може мати значний регуляризаційний ефект для моделі (і в меншій мірі, що може поліпшити оптимізацію). По-друге, воно використовує більш загальну ідею, що вивчення розподілу вхідних даних може допомогти дізнатися про відображення від входів до виходів.

Обидві ці ідеї передбачають багато складних взаємодій між декількома частинами алгоритму машинного навчання, які не цілком зрозумілі.

Перша ідея полягає в тому, що вибір початкових параметрів для глибокої нейронної мережі може мати сильний регуляризаційний ефект на його продуктивність – це є найменш зрозумілим. У той час, коли попередня підготовка стала популярною, це було зрозуміло як ініціалізація моделі в тому місці, що призведе до наближення до одного місцевого мінімуму, а не іншого. Сьогодні місцеві мінімуми більше не вважаються серйозною проблемою для оптимізації нейронної мережі. Тепер ми знаємо, що наша стандартна процедура навчання нейронних мереж

звичайно не знаходить критичну точку будь-якого роду. Можливо, що попередня підготовка ініціалізує модель в тому місці, яке інакше було б недоступним - наприклад, регіон, оточений районами, де функція вартості настільки варіює від одного прикладу до іншого, що мініатюри дають лише дуже шумну оцінку градієнта, або регіон, оточений районами, де матриця Гессіана настільки погано обумовлена, що методи градієнтного спуску повинні робити дуже маленькі кроки. Проте наша здатність точно визначати, які аспекти попередньо підготовлених параметрів зберігаються під час контрольного етапу навчання, є обмеженим. Це одна з причин того, що сучасні підходи зазвичай використовують одночасно навчання без учителя та навчання з учителем, а не два послідовні етапи. Можна також уникати боротьби з цими складними уявленнями про те, як оптимізація на контрольному етапі навчання зберігає інформацію від непідконтрольної стадії навчання, просто заморожуючи параметри для екстракторів ознак та використовуючи контрольоване навчання, лише для того, щоб додати класифікатор на додаток до вивчених ознак.

Інша ідея, що алгоритм навчання може використовувати інформацію, отриману на етапі без учителя, для кращого функціонування на етапі навчання з учителем, краще зрозуміла. Основна ідея полягає в тому, що деякі ознаки, корисні для безконтрольного завдання, також можуть бути корисними для завдання навчання з учителем. Наприклад, якщо ми тренуємо генеративну модель зображень автомобілів і мотоциклів, то потрібно знати про колеса та про те, скільки коліс має бути на зображенні. Якщо нам пощастило, то представлення коліс буде прийнято у формі, яку користувач, до котрого навчається, може легко отримати доступ. Це ще не зрозуміло на математичному, теоретичному рівні, тому не завжди можна передбачити, які завдання будуть корисні від навчання без учителя в такий спосіб. Багато аспектів цього підходу дуже залежать від конкретних моделей, що використовуються. Наприклад, якщо ми хочемо додати лінійний класифікатор поверх попередньо натренованих ознак, ознаки повинні зробити основні класи лінійно відокремлюваними. Ці властивості часто бувають природними, але це не завжди відбуваються. Це ще одна причина того, що одночасне контрольоване та

безконтрольне навчання може бути кращим - обмеження, що накладаються вихідним рівнем, природно включаються з самого початку.

З точки зору безконтрольного попереднього навчання, як вивчення представлення, ми можемо очікувати, що тренування без учителя буде ефективнішим, коли початкове представлення є поганим. Одним з ключових прикладів цього є використання векторного представлення слів. Слова, представлені векторами, не дуже інформативні, оскільки кожні два окремі вектори є на однаковій відстані один від одного (квадратична відстань L^2 від 2). Навчені векторні представлення слів, природно, кодують схожість між словами за їх відстанню один від одного. Через це безконтрольне попереднє навчання особливо корисно при обробці слів. Це менш корисно при обробці зображень, можливо тому, що зображення вже лежать у багатому векторному просторі, де відстані забезпечують низькоякісну метрику подібності.

З точки зору безконтрольного попереднього навчання як регуляризатора, ми можемо очікувати, що безконтрольне попереднє тренування буде найбільш корисним, коли кількість розмічених прикладів дуже мала. Перевага напівавтоматичного навчання шляхом безконтрольного попереднього тренування з безліччю нерозмічених прикладів та декількома поміченими прикладами була особливо ясною в 2011 році з безконтрольним попереднім навчанням, коли було виграно два міжнародних навчальних змагань з навчання через перенос (transfer learning), в налаштування, де кількість наведених прикладів у цільовому завданні була мала (від кількох до десятків прикладів на один клас). Ці ефекти також були задокументовані у ретельно перевірених експериментах.

Інші фактори можуть бути задіяні. Наприклад, неконтрольоване попереднє навчання, ймовірно, буде найбільш корисним, коли функція, яку потрібно вивчати, є надзвичайно складною. Неконтрольований навчальний процес відрізняється від регуляризаторів, таких як weight decay, тому що він не зміщує учня на пошук простої функції, а на виявлення ознак функцій, корисних для безконтрольного навчального завдання. Якщо справжні основні функції ускладнені та сформовані

закономірностями розподілу вхідних даних, неконтрольоване навчання може бути більш корисним регуляризатором.

Ми зараз аналізуємо деякі випадки, коли попереднє тренування без учителя, як відомо, сприяло покращенню, та пояснюємо, що відомо про те, чому це покращення відбувається. Безконтрольне попереднє тренування зазвичай використовується для покращення класифікаторів, і, як правило, найбільш цікаво з точки зору зменшення помилки тестового набору. Тим не менш, безумовне попереднє навчання може допомогти з іншими завданнями, не тільки класифікацією, і може діяти для поліпшення оптимізації, а не лише для регуляризації. Наприклад, воно може зменшити як помилку на тренувальному, так і на тестовому наборі для глибоких автоенкодерів.

Було здійснено багато експериментів, щоб пояснити кілька успішних випадків використання попереднього тренування без учителя. Обидва покращення навчальної помилки та покращення тестової помилки можна пояснити з точки зору безумовного попереднього тренування, беручи параметри в регіон, який інакше було б недоступним. Навчання нейронної мережі є недетермінованим, і сходиться до іншої функції кожного разу, коли воно запускається. Тренування може зупинитися в точці, де градієнт стає невеликим, точкою, коли раннє припинення (early stopping) закінчує навчання, щоб запобігти перенавчанню, або в точці, де градієнт є великим, але важко знайти крок униз через проблеми, такі як стохастичність або погане кондиціонування Гессіана. Нейронні мережі, які попередньо натреновані без учителя, постійно зупиняються в тому ж регіоні функціонального простору, тоді як нейронні мережі без попереднього тренування послідовно зупиняються в іншому регіоні. Регіон, куди прибули попередньо натреновані мережі, є меншим, що свідчить про те, що попередня підготовка зменшує дисперсію процесу оцінювання, що, у свою чергу, може зменшити ризик серйозного перенавчання. Іншими словами, попереднє тренування без учителя ініціалізує параметри нейронної мережі в область, в якій вони не виходять, а результати, що слідують за цією ініціалізацією, є більш послідовними та менш ймовірними, що вони будуть дуже поганими, ніж без цієї ініціалізації.

Важливим питанням є те, як безконтрольне попереднє тренування може діяти як регуляризатор. Одна гіпотеза полягає в тому, що попередня підготовка заохочує алгоритм навчання виявляти ознаки, пов'язані з основними причинами, які генерують спостережувані дані.

У порівнянні з іншими формами навчання без учителя, безконтрольне попереднє тренування має недолік, оскільки воно працює з двома окремими фазами навчання. Багато стратегій регуляризації мають перевагу, що дозволяють користувачу контролювати силу регуляризації шляхом коригування значення певного гіперпараметра. Попереднє тренування без учителя не дає чіткого способу регулювати силу регуляризації, що виникає після етапу навчання без учителя. Натомість існує дуже багато гіперпараметрів, вплив яких можна виміряти після того, як це сталося, але часто це важко передбачити заздалегідь. Коли ми здійснюємо безконтрольне та контрольоване навчання одночасно, замість того, щоб використовувати стратегію попереднього тренування, існує єдиний гіперпараметр, зазвичай коефіцієнт приєднання до безконтрольної вартості, що визначає, наскільки сильно безконтрольна мета буде регулювати контрольовану модель. Завжди можна передбачити меншу регуляризацію, зменшивши цей коефіцієнт. У випадку безконтрольного попереднього навчання не існує способу гнучкої адаптації сили регуляризації - або контрольована модель ініціалізується на попередньо підготовлених параметрах, або ні.

Інший недолік двох окремих стадій навчання полягає в тому, що кожен етап має свої гіперпараметри. Продуктивність другої фази, як правило, не може бути передбачена на першому етапі, тому існує довга затримка між пропозицією гіперпараметрів для першої фази та можливістю їх оновлення за допомогою зворотного зв'язку з другої фази. Найбільш принциповим підходом є використання помилки встановлення валідації на контрольній фазі для вибору гіперпараметрів фази попередньої тренування. На практиці, деякі гіперпараметри, як і кількість ітерацій, що передують тренуванню, зручніше встановлювати під час фази попереднього тренування, використовуючи раннє зупинення на безконтрольній цілі, що не є

ідеальним, але обчислювально значно дешевшим, ніж використання цілі контрольованого навчання.

Сьогодні безумовне попереднє навчання було переважно відкинуте, за винятком обробки природної мови, де природне представлення слів як one-hot векторів не містить інформації про схожість та де наявні дуже великі нерозмічені набори. У цьому випадку перевага попереднього тренування полягає в тому, що один раз можна перегнати величезний нерозмічений набір (наприклад, з корпусом, що містить мільярди слів), навчитися доброму представленню (як правило, слова, а також речення), а потім використовувати це представлення або підлаштувати його для контрольованого завдання, для якого навчальний набір містить набагато менше прикладів.

2.3 Висновки за розділом

Під попереднім тренуванням розуміють деяке навчання мережі перед основним її навчанням. Методи попереднього тренування створені, щоб покращити показники нейронної мережі, яку планується навчати.

В цьому розділі були розглянуті методи попереднього навчання, які базувались як на алгоритмах навчання без вчителя так і на алгоритмах навчання з вчителем. Після попереднього тренування нейронна мережа може швидше навчатись, мати кращу точність або бути менш схильною до проблеми перенавчання. Існуючі методи зазвичай базуються на жадібних алгоритмах, які виконують пошарове навчання.

Безконтрольне попереднє навчання поєднує в собі дві різні ідеї. По-перше, воно використовує ідею, що вибір початкових параметрів для глибокої нейронної мережі може мати значний регуляризаційний ефект для моделі. По-друге, воно використовує більш загальну ідею, що вивчення розподілу вхідних даних може допомогти дізнатися про відображення від входів до виходів.

3 ПОПЕРЕДНЄ ТРЕНУВАННЯ ЗА ДОПОМОГОЮ СПАЙКОВИХ НЕЙРОННИХ МЕРЕЖ

Багато методів розроблено для попереднього навчання штучних нейронних мереж. В даній роботі представлений новий підхід, який базується на використанні спайкових нейронних мереж в попередньому навчанні.

Даний підхід демонструється на прикладі задачі класифікації.

3.1 Розробка методу

Ідея базується на нашому розумінні того, як людина може впізнавати закономірності, як маркування даних працює в мозку. Спочатку людина просто спостерігає за світом: навчається бачити, чути і т. п.. Спостерігаючи різні речі і відмінності між ними, виробляються критерії (атрибути), такі як колір, розмір, форма та ін.. Комбінація атрибутів, що істотно відрізняє один об'єкт відмінні від інших, використовується для маркування; атрибути, що мають невелику різницю, ігноруються (абстрагуються). Зустріч з об'єктами з певними значеннями атрибутів, що спостерігалися раніше, дозволяє мозкові розпізнавати назви цих об'єктів.

Штучні нейронні мережі дійсно хороші в останній частині, в співставленні відповідних значеннях атрибутів з назвами (мітками). Вони добре працюють з позначеними даними (контрольоване навчання); однак, вони не настільки хороші в обробці нерозмічених даних, що може бути дуже корисним. З іншого боку, головний мозок (біологічні нейронні мережі), кращий в першій частині, коли йдеться про нерозмічені дані, коли людина просто спостерігає за світом і виробляє різні критерії. Було вирішено отримати вигоду від обох рішень, тому об'єднано два типи нейронних мереж: SNN для навчання на нерозмічених даних та ANN для навчання на розмічених. Багато аспектів повинні бути переглянуті, щоб об'єднати такі різні типи нейронних мереж. Мережі працюють з різними типами об'єктів, і здається, що єдиним спільним є роль ваг, тобто наскільки сильно один нейрон впливає на інший.

Отже, потрібно побудувати спайкову нейронну мережу [28], натренувати її за допомого безконтрольного правила навчання, перенести навчене знання в штучну нейронну мережу та натренувати її використовуючи вже розмічені дані.

Для простоти розглянемо спайкову нейронну мережу, яка складається лише з вхідного та збуджувального (вихідного) шарів (рис. 3.1). Кожен нейрон вхідного шару сполучений з кожним нейроном збуджувального шару.

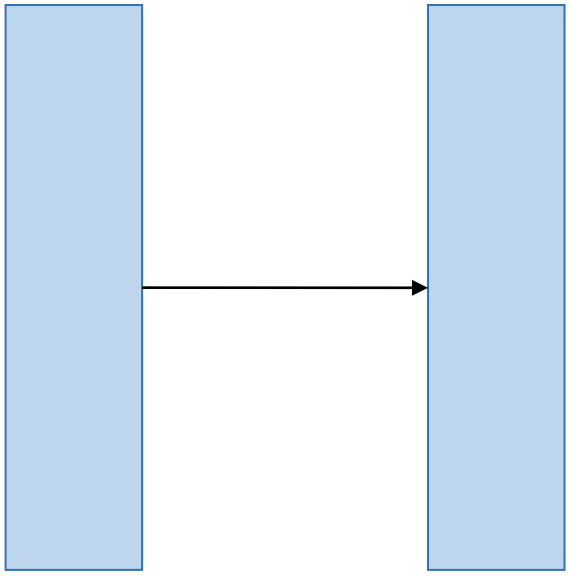

Рисунок 3.1 Стркуктура одношарової спайкової нейронної мережі

Побудована нейронна мережа повинна потім навчатись за допомогою безконтрольного правила навчання, такого як STDP. Так як навчання безконтрольне, то значення класів (позначення) до мережі не подаються.

Після навчання нейронам збуджувального шару присвоюються класи на основі їх найвищої середньої відповіді на класи при подачі тренувального набору. В підсумку, за кожний клас буде відповідати декілька нейронів. Це єдиний крок, де подаються позначення, але це відбувається після навчання, тому саме навчання відбувається без учителя.

Після завершення етапу попереднього навчання SNN замінюється на ANN, потрібно використати навчене знання в штучній нейронній мережі та підлаштувати (навчити) її за допомогою розмічених даних. SNN та ANN - це різні види нейронних

мереж, які працюють з різними типами об'єктів (спайкова нейронна мережа використовує імпульси, дискретні події; штучна нейронна мережа використовує числа, неперервні значення), і в даному випадку виконують різні типи навчання. Тому для трансферу моделі недостатньо простого копіювання ваг, хоча вони відіграють ту ж саму роль. Тому потрібно розглянути ще декілька частин, які треба змінити, для перенесення знання.

При навчанні штучних нейронних мереж з учителем нейрони вихідного шару зазвичай відповідають тільки за певний клас: коли на вхід до мережі подаються деякі дані, то вихідний нейрон того класу, до якого ці дані належать повинен видати найвище значення. В спайковій нейронній мережі відсутній такий шар, а вихідний (збуджувальний) шар містить по декілька нейронів, які відповідають за певний клас на основі середнього значення. За який саме клас вони відповідають – це визначалось на етапі після навчання спайкової мережі. Тому в штучній нейронній мережі, до структури спайкової мережі потрібно додати ще один шар на виході та з’єднати його з попереднім шаром так, щоб за певний клас відповідав лише один вихідний нейрон. Отримана структура штучної нейронної мережа зображена на рисунку 3.2.

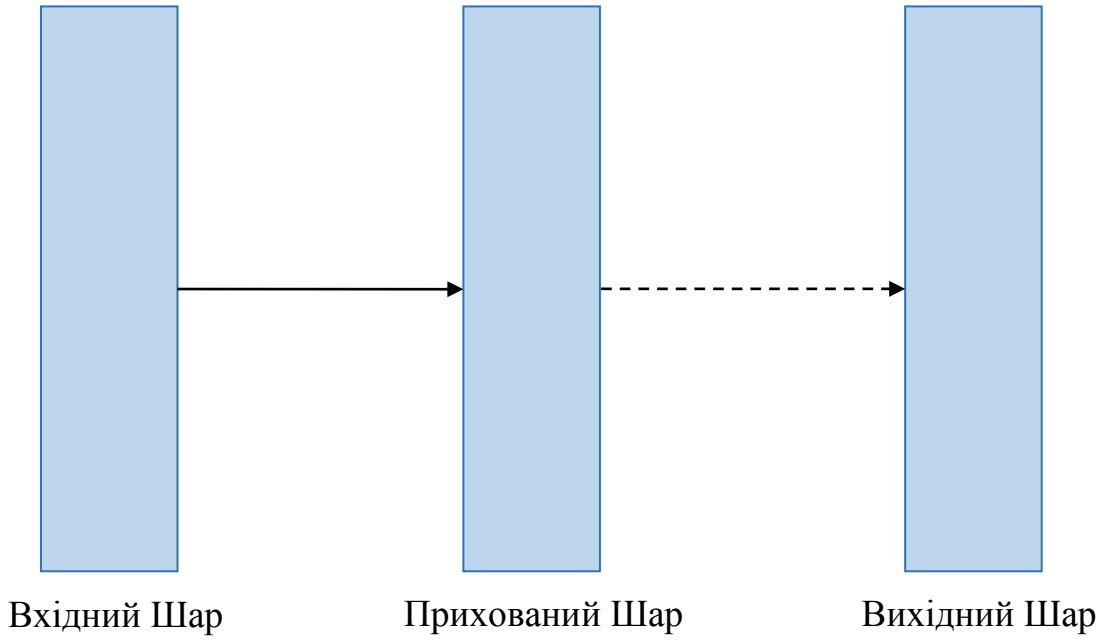

Рисунок 3.2 - Структура штучної нейронної мережі

Отже, прихований шар нової мережі - це збуджувальний шар спайкової нейронної мережі. Так само як і в спайковій нейронній мережі, тут кожен нейрон вхідного шару сполучений з кожним нейроном прихованого шару. Для сполучення прихованого шару з вихідним використовуються позначення збуджувальних нейронів спайкової нейронної мережі, яке відбулося після її навчання. Нейрони прихованого шару, які відповідають за один і той самий клас з’єднуються з одним нейроном вихідного шару (рис. 3.3). Тобто, роль нового вихідного шару в тому, щоб об’єднати декілька нейронів одного класу в один, роблячи вихідний шар придатним для контрольованих алгоритмів навчання штучних нейронних мереж. Ці зв’язки можуть бути фіксованими, так як, по суті, вони грають роль відображення.

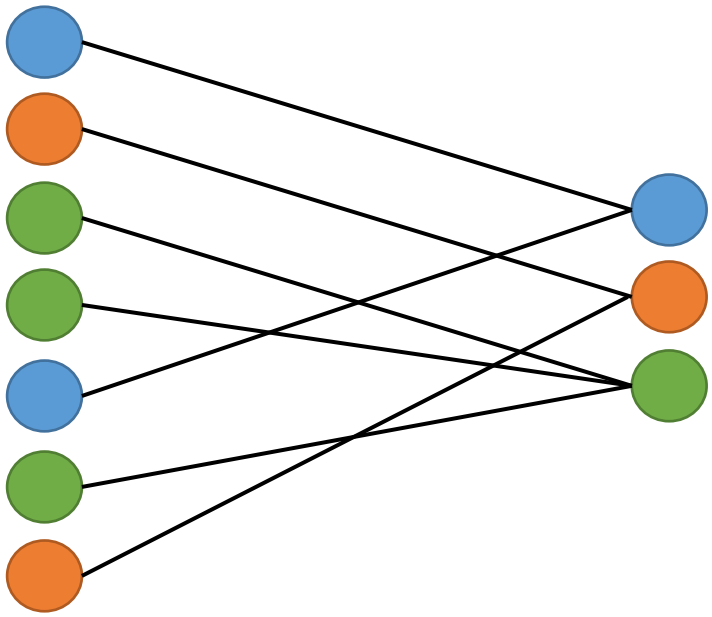

Рисунок 3.3 - Приклад з'єднання нейронів прихованого шару (розміченими на етапі після тренування спайкової нейронної мережі) з нейронами вихідного шару

Нейрони спайкової нейронної мережі відрізняються від нейронів класичної штучної нейронної мережі, тому при зміні типу мережі потрібно змінити і тип нейронів. Зазвичай в штучних нейронних мережах використовують нейрони, які

просто сумують вхідні дані, застосовують деяку активаційну функцію та виробляють число. В ANN часто використовують Sigmoid (або її узагальнення Softmax) для вихідного шару та ReLU для прихованих шарів, але й можуть використовуватись і інші. Нейрони спайкових мереж є складними одиницями, що випускають імпульси при певних умовах; як правило, ці умови описуються диференційними рівняннями.

Ваги відіграють ключову роль в навчанні нейронних мереж, адже якраз вони є відображенням навченого знання. Навчене знання спайкової мережі, яку ми розглядаємо міститься у вагах між вхідним та збуджувальним шарами. В штучній мережі воно міститься в вагах між вхідним та прихованим шарами. Ваги між прихованим та вихідним шарами, які розглядались раніше, задаються для об’єднання декількох нейронів позначаючих один клас в один нейрон, по суті, грають роль відображення. Потрібно перенести ваги між вхідним та збуджувальним шарами зі спайкової мережі в ваги між вхідним та прихованим шарами штучної нейронної мережі. Проблема в тому, що в ці мережі сприймають ваги по-різному (значення, дисперсія, шум та інше), проте роль ваг залишається однаковою. Для того, щоб зрозуміти як спайкова нейронна мережа сприймає ваги натренуємо її на наборі рукописних цифр та візуалізуємо ваги (рис. 3.4). Те ж саме зробимо для штучної нейронної мережі, для ваг між вхідним та прихованим шарами, не копіюючи їх зі спайкової мережі. Але з метою візуалізації тих же класів, ваги між прихованим та вихідним шарами візьмемо на основі розмітки збуджувальних нейронів, отриманої після тренування спайкової мережі. Візуалізація ваг штучної нейронної мережі між вхідним та прихованим шарами показана на рисунку 3.5.

Рисунок 3.4 - Візуалізація натренованих ваг (між вхідним та збуджувальним шарами) спайкової нейронної мережі

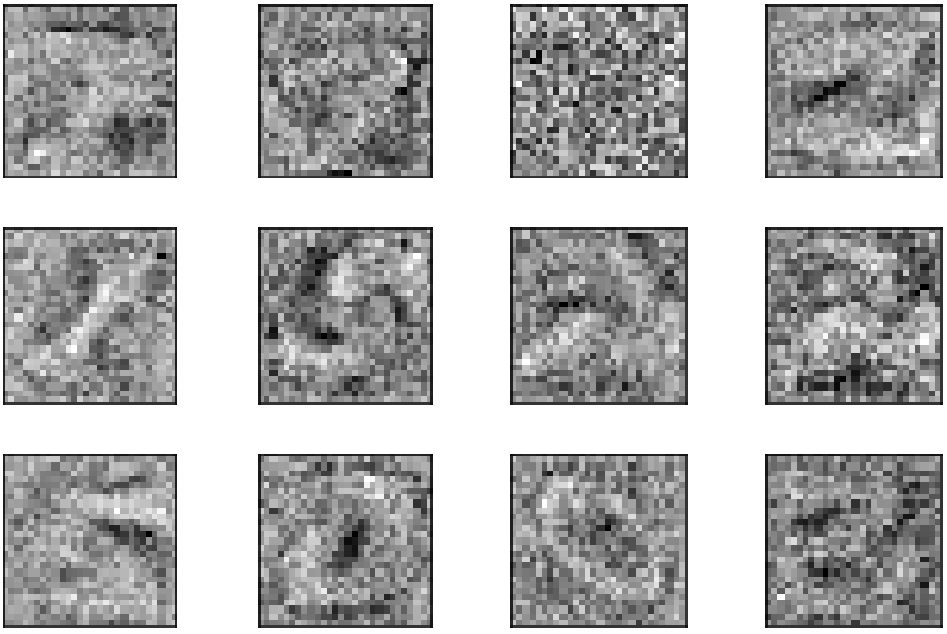

Рисунок 3.5 - Візуалізація натренованих ваг (між вхідним та прихованим шарами) штучної нейронної мережі

Треновані ваги спайкової мережі повинні більше виглядати як треновані ваги штучної нейронної мережі, але без попереднього навчання. Для цього їх потрібно

модифікувати, щоб штучна нейронна мережа сприйняла їх як свої власні, коли вони будуть скопійовані. Використаємо декілька простих стратегій для цього: середня нормалізація, нормалізація та стандартизація (рівняння (3.1) - (3.3) відповідно).

$$W_{mean\ normalized} = W - \mu \tag{3.1}$$

$$W_{nomalized} = \frac{W - W_{min}}{W_{max} - W_{min}} \tag{3.2}$$

$$W_{standardized} = (W - W_{min})/\sigma \tag{3.3}$$

Візуалізуємо ваги спайкової нейронної мережі після відповідних модифікацій (рис. 3.6 - 3.8).

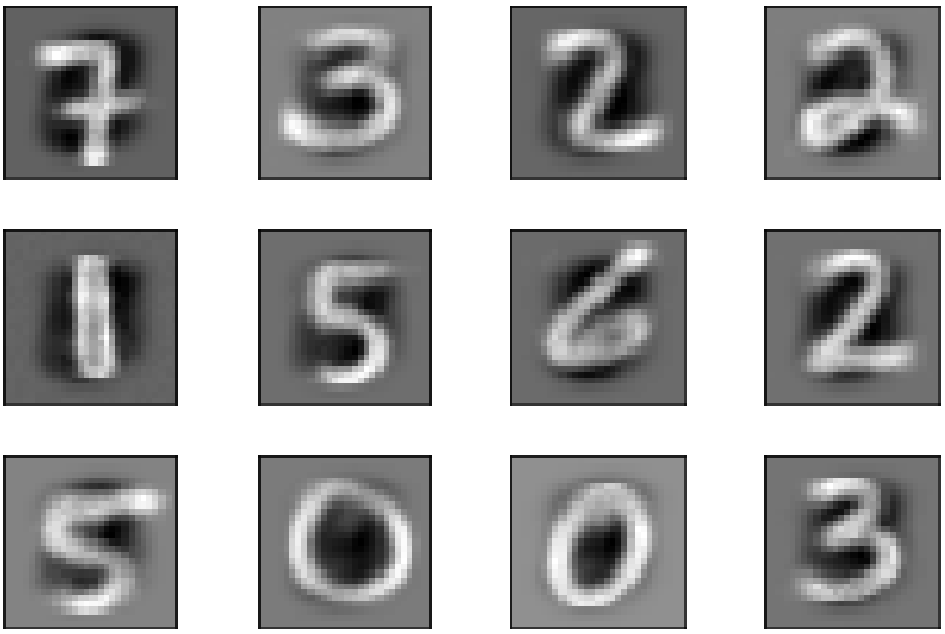

Рисунок 3.6 - Візуалізація натренованих ваг спайкової нейронної мережі після середньої нормалізації

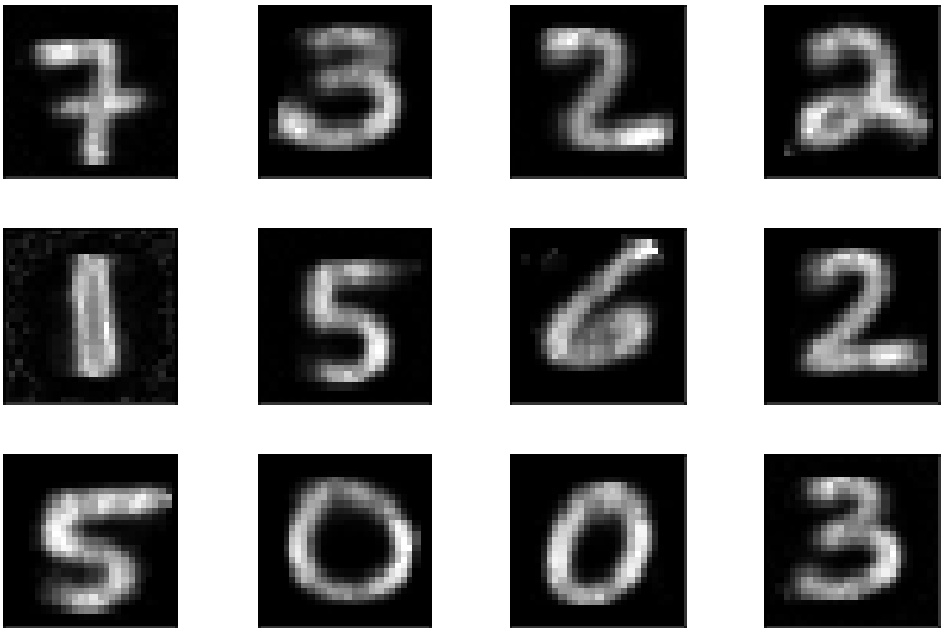

Рисунок 3.7 - Візуалізація натренованих ваг спайкової нейронної мережі після нормалізації

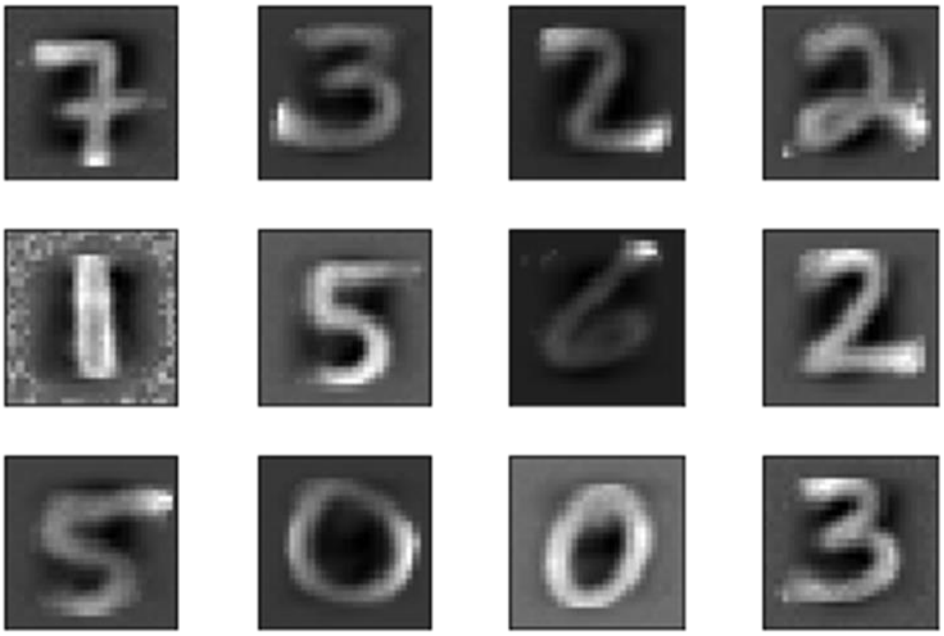

Рисунок 3.8 - Візуалізація натренованих ваг спайкової нейронної мережі після стандартизації

Ваги спайкових нейронних мереж стають більш придатними для звичайних штучних нейронних мереж після простого масштабування. Ці стратегії були вибрані

з розрахунком на те, що штучні нейронні мережі навчаються краще, коли початкові ваги масштабовані (нормалізовані) певним чином [29].

Більш складні стратегії можуть включати в себе додавання шуму, спотворення тощо. Ідеальний алгоритм перенесення ваг повинен робити початкову точність штучної мережі після перенесення ваг такою ж, як точність щойно навченої спайкової мережі.

Після відповідних модифікацій спайкової мережі можна починати тренувати штучну нейронну мережу за допомогою алгоритмів заснованих на градієнтному спуску або інших.

Важливою частиною є також верифікація методу, тобто перевірка того, чи дійсно штучна нейронна мережа використовує певне знання, отримане за допомогою спайкової нейронної мережі або її ваги лише непогано ініціалізовані з величинами непоганого масштабу. Таку перевірку зробити досить просто: достатньо змінити позначення збуджуючих нейронів, отримане на етапі після тренування спайкової нейронної мережі, перемішавши їх випадковим чином. Якщо після цього штучна мережа буде видавати таку ж точність, яку видавала з правильними позначеннями, то це значить, що перенесене зі спайкової нейронної мережі знання - це лише непогано ініціалізовані ваги з правильним масштабом, а ніяк не певне натреноване знання. В іншому випадку це означає, що метод дійсно працює і штучна мережа сприймає та використовує знання спайкової нейронної мережі.

3.2 Реалізація методу та проведення експериментів

3.2.1 Набір даних

Для простоти побудови нейронної мережі була обрана задача класифікації рукописних цифр та використаний набір даних MNIST. Він містить 60 000 навчальних прикладів та 10 000 прикладів тестування. При вирішенні такого завдання важко буде побачити переваги використання спайкової мережі на етапі попереднього тренування, тому було вирішено ускладнити задачу зробивши навчання напівавтоматичним.

Для напівавтоматичного навчання кількість навчальних прикладів має бути малою. У даній роботі для навчання використовуються лише 600 позначених прикладів з тренувального набору, решта навчального набору використовується як нерозмічена. Для вимірювання точності використовується 10 000 прикладів тестового набору.

3.2.2 Вибір структур та параметрів

Структура та параметри спайкової нейронної мережі взяті з [30], де використовують спайкову нейронну мережу для розпізнавання цифр: модель нейронів - Leaky Integrate-and-Fire; правило навчання - STDP; наявні вхідний та обробний шари, які складаються з 784 та 400 нейронів відповідно. Обробний шар складається з збуджувальних і гальмівних зв’язків, тому їх можна розглядати як два окремих шари (рис. 3.9).

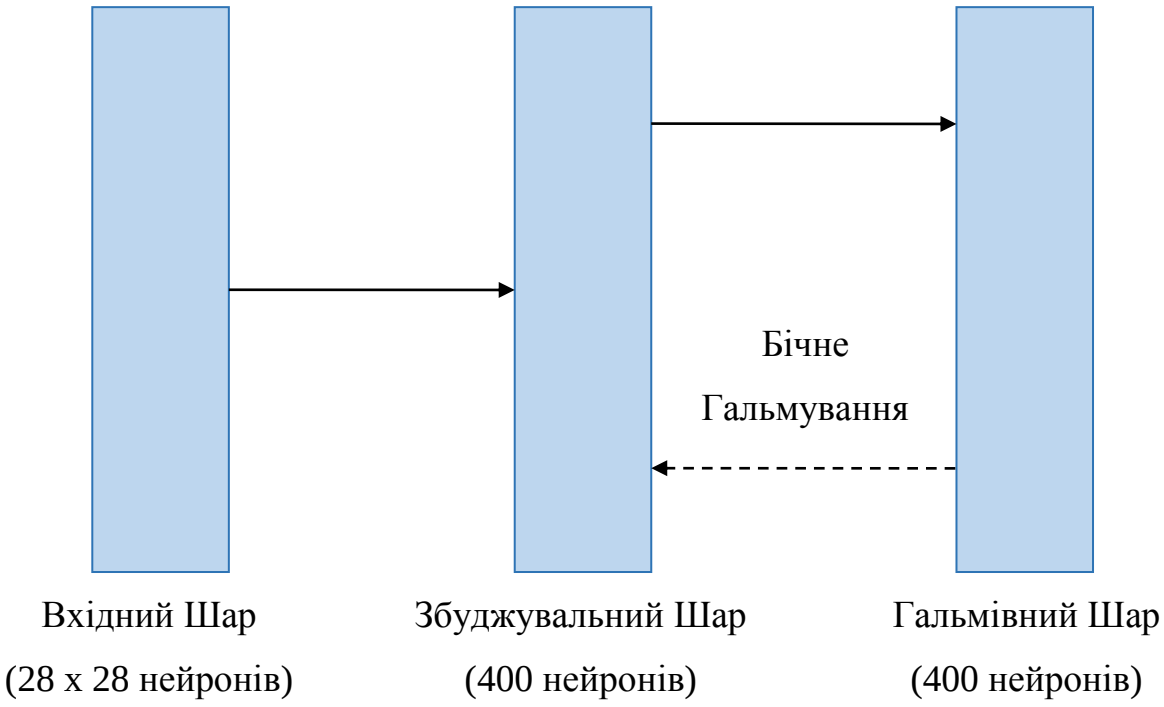

Рисунок 3.9 - Структура спайкової нейронної мережі

Збуджувальні нейрони взаємно пов'язані з гальмівними нейронами. Кожен гальмівний нейрон пов'язаний з усіма збуджувальними нейронами, крім того, від

якого він отримує з'єднання. Ця структура дозволяє імітувати процес, відомий як бічне гальмування (lateral inhibition), здатність збуджуваного нейрона знизити активність сусідніх нейронів. Вихідний шар, який виконує класифікацію, відсутній, оскільки навчання без учителя.

Схожа структура використовується і для штучної нейронної мережі, але з модифікаціями щодо іншого типу мережі та контрольованого типу навчання. Для виконання навчання мережі з учителем введений додатковий шар (рис. 3.10). З’єднання між прихованим та вихідним шарами побудовані за принципом, описаним в попередньому підпункті: присвоєння класів збуджувальних нейронів спайкової мережі використовується для побудови фіксованих ваг між прихованим та вихідним шарами ANN, тобто, нейрон прихованого шару, якому призначений певний клас в SNN в збуджувальному шарі, підключається тільки до вихідного нейрону відповідного класу. Підключення, в даному випадку, означає встановлення значення ваг в 1. Гальмівний шар, який використовувався в спайковій нейронній мережі проігнорований.

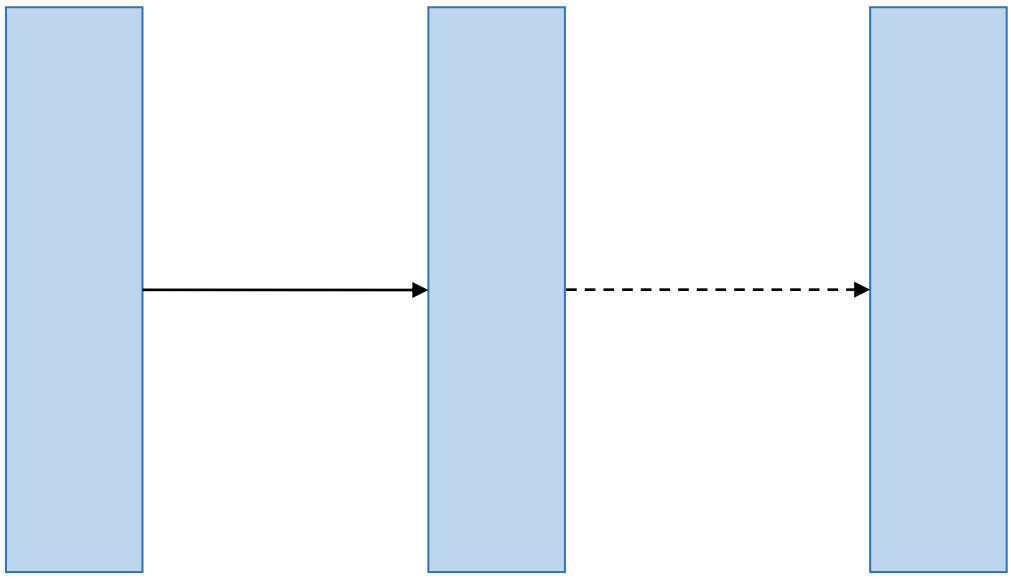

Рисунок 3.10 - Структура штучної нейронної мережі

Спайкові нейрони замінюються в ANN на штучні. Такі функції активації як Sigmoid, Tanh та ReLU використовуються далі в експериментах в прихованому шарі. У нейронах вихідного шару використовується функція активації Softmax. Для обробки ваг при їх перенесенні в експериментах використовується середня нормалізація, нормалізація та стандартизація. В якості алгоритму навчання з учителем обрано Adam, градієнтний алгоритм оптимізації.

В експериментах, де не використовуються попередньо натреновані ваги, ваги ініціалізуються за допомогою випадкового нормального розподілу з середнім 0 та стандартним відхиленням $\frac{1}{\sqrt{\text{кількість входів до нейрону}}}$, тобто $\frac{1}{28}$.

Оскільки дуже невелика кількість даних використовується для навчання ANN, у всіх експериментах використовується додатковий метод регуляризації, 50% dropout для прихованого шару.

3.2.3 Проведення експериментів

Першим кроком (попереднє навчання без учителя) є навчання спайкової нейронної мережі. На цьому етапі використовуються всі 60 000 навчальних зображень (без міток). Після тренування мережі, мітки призначаються нейронам збуджуючого шару на основі середньої відповіді на клас цифр, використовуючи 600 позначених прикладів з тренувального набору.

Другий крок полягає в тому, щоб перенести натреноване знання з SNN до ANN.

Останній крок - навчання штучної нейронної мережі. Для проведення навчання ANN з учителем використовуються 600 позначених прикладів з тренувального набору.

Для визначення точності використовується формула (3.4). Прогнозування відбувається на тестовому наборі з 10 000 прикладів.

$$\text{точність} = \frac{\text{кількість правильних прогнозів}}{\text{кількість прогнозів}} \tag{3.4}$$

У цих експериментах ми зосереджені на спостереженні за впливом використання попереднього тренування з SNN, впливом використання різних стратегій перенесення ваг (без зміни, середньої нормалізації, нормалізації та стандартизації) та різних функцій активації (Sigmoid, Tanh, ReLU) в нейронах прихованого шару.

Для початку протестуємо нейронні мережі, де, в якості активаційної функції нейронів прихованого шару взято функцію Sigmoid. Результати експериментів відображені в таблиці 3.1. Попереднє тренування за допомогою спайкової нейронної мережі дещо покращило точність штучної нейронної мережі при використанні середньої нормалізації при перенесенні ваг (з 88.19% до 89.35%), та майже не покращило при використанні стандартизації (88.33%). При відсутності обробки ваг при їх перенесенні та при використанні нормалізації результат погіршився.

Таблиця 3.1 - Результати експериментів з використанням активаційної функції Sigmoid

Попереднє навчання за допомогою SNN	Стратегія перенесення ваг	Точність (%)
-	-	88.19
+	-	87.76
+	Середня нормалізація	89.35
+	Нормалізація	88.09
+	Стандартизація	88.33

Результати тестування нейронних мереж, де, в якості активаційної функції нейронів прихованого шару взято функцію Tanh відображені в таблиці 3.1. Попереднє тренування за допомогою спайкової нейронної мережі практично не покращило показників штучної нейронної мережі (при середній нормалізації з 87.08% до 87.72%).

Таблиця 3.2 - Результати експериментів з використанням активаційної функції Tanh

Попереднє навчання за допомогою SNN	Стратегія перенесення ваг	Точність (%)
-	-	87.08
+	-	86.51
+	Середня нормалізація	87.72
+	Нормалізація	85.94
+	Стандартизація	86.57

Найкращі результати показали мережі з активаційною функцією ReLU (таблиця 3.3). Використання спайкових нейронних мереж на етапі попереднього тренування покращило точність штучної нейронної мережі з 88.66% до 91.08% (при середній нормалізації ваг) та до 91.02% (при стандартизації ваг). Візуалізації ваг цих мереж зображені на рисунках 3.11 - 3.13 відповідно.

Таблиця 3.3 - Результати експериментів з використанням активаційної функції ReLU

Попереднє навчання за допомогою SNN	Стратегія перенесення ваг	Точність (%)
-	-	88.66
+	-	87.51
+	Середня нормалізація	91.08
+	Нормалізація	87.76
+	Стандартизація	91.02

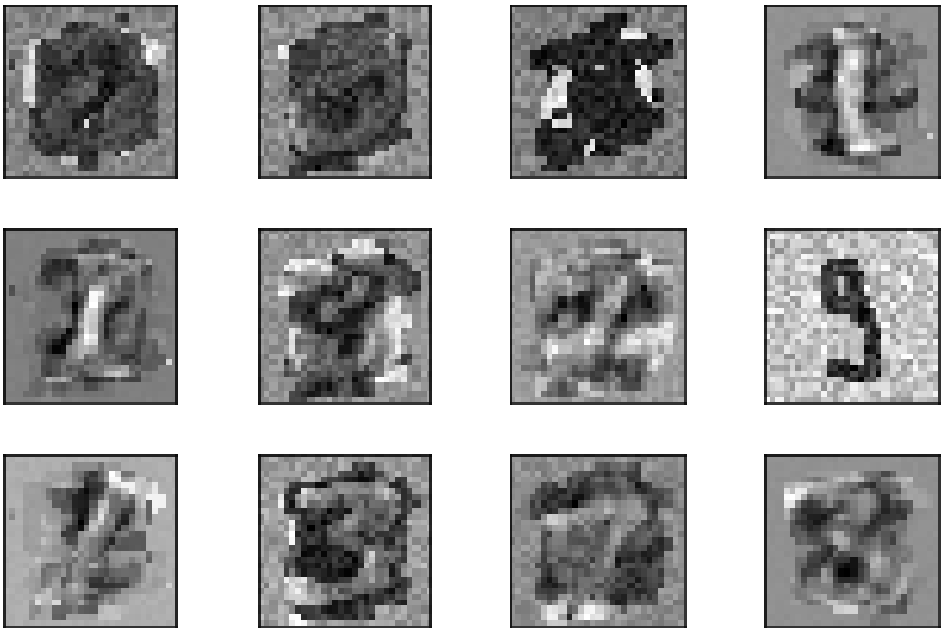

Рисунок 3.11 - Візуалізація ваг після тренування ANN з нуля з функцією активації ReLU в нейронах прихованого шару

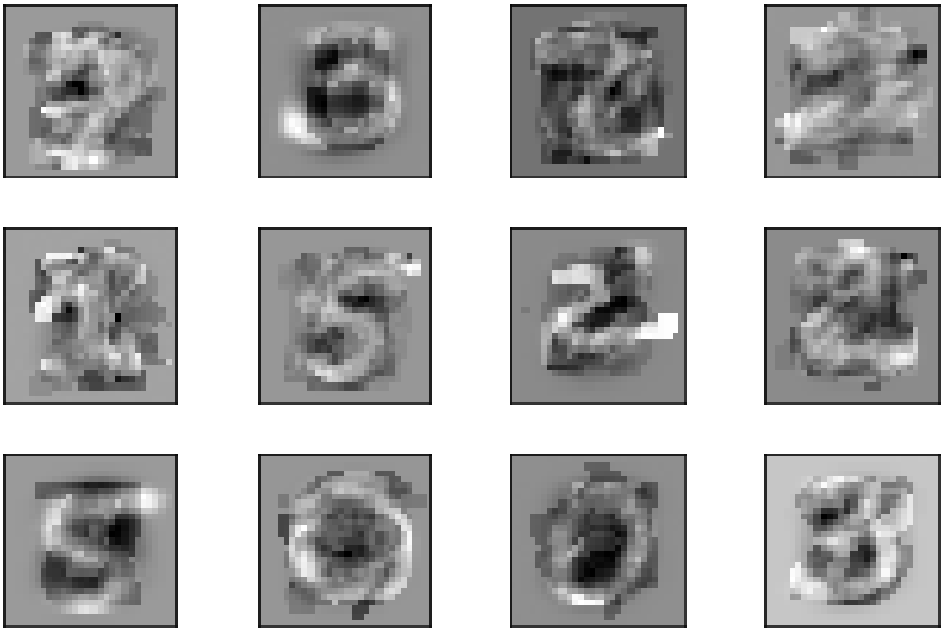

Рисунок 3.12 - Візуалізація ваг після попереднього тренування за допомогою SNN, середньої нормалізації та фінального тренування ANN з функцією активації ReLU в нейронах прихованого шару

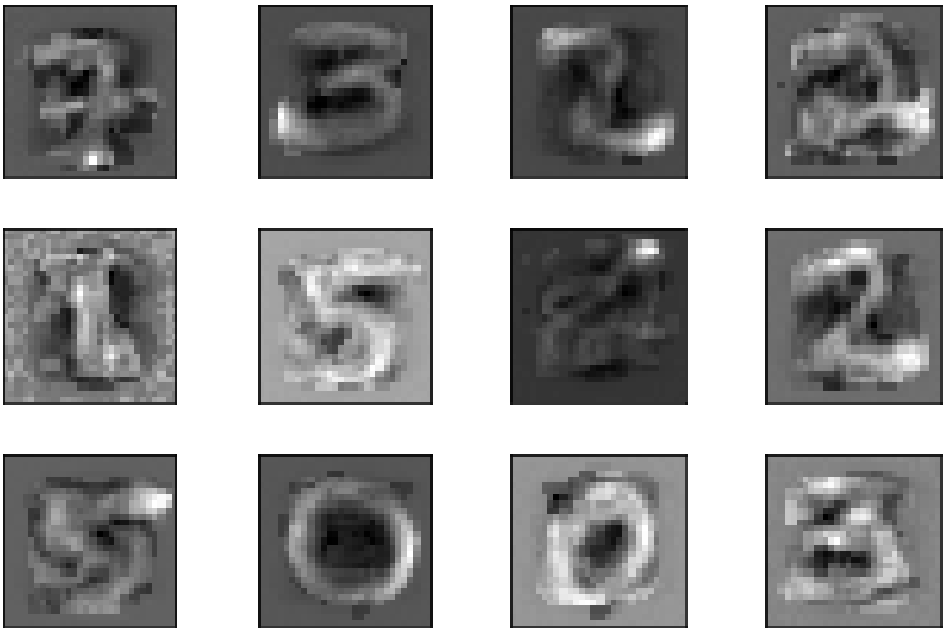

Рисунок 3.13 - Візуалізація ваг після попереднього тренування за допомогою SNN, стандартизації та фінального тренування ANN з функцією активації ReLU в нейронах прихованого шару

Якщо порівнювати ваги на рисунку 3.6 та 3.8 з вагами, показаними на рисунках 3.12 та 3.13, то вони стали більш спотвореними та менш контрастними після навчання ANN.

Для того, щоб перевірити, чи дійсно штучна нейронна мережа використовує певне знання, отримане за допомогою спайкової нейронної мережі змінимо позначення збуджуючих нейронів, отримане на етапі після тренування спайкової нейронної мережі, перемішавши їх випадковим чином. Дане позначення нейронів використовується для створення зв’язків між прихованим та вихідним шарами штучної нейронної мережі. Спроба натренувати дану мережу ні до чого не привела, точність такої мережі склала 11.3%. Це означає, що метод дійсно працює і штучна мережа сприймає та використовує знання спайкової нейронної мережі.

3.3 Аналіз отриманих результатів

Краща ANN без етапу попереднього тренування з SNN досягла точності 88,66%, вона використовувала функцію активації ReLU. Використання попередньо натренованих ваг SNN без масштабування не покращило точність класифікатора, оскільки ANN не могла правильно сприйняти ваги, тому штучна нейронна мережа почала навчання як з самого початку, з погано ініціалізованими вагами. Використання середньої нормалізації та стандартизації дозволило ANN сприйняти передані ваги як свої власно натреновані, отже, підвищилася точність (при використанні функцій активації Sigmoid та ReLU). При використанні функції активації Tanh з середньою нормалізацією ваг, мережа отримала трохи кращу точність, ніж мережа без попереднього навчання. Проста нормалізація попередньо натренованих ваг не покращила точність роботи мережі.

Отже, штучні нейронні мережі, що використовували функцію активації ReLU досягли кращих результатів. При перенесенні ваг зі спайкової нейронної мережі на штучну нейронну мережу використання середньої нормалізації та стандартизації найкраще вплинули на подальше навчання штучної нейронної мережі та покращило її точність з 88.66% до 91.08% та 91.02%.

3.4 Висновки за розділом

В даному розділі був реалізований новий метод попереднього тренування штучної нейронної мережі, що базується на використанні спайкової нейронної мережі для задачі напівавтоматичного навчання. Були обрані компоненти методу, які можуть впливати на показники навчання штучної нейронної мережі та обрані ті конфігурації, які найкраще підвищили точність навчання.

Також при проведенні експериментів була зроблена перевірка, яка дозволила впевнитись, що даний метод спрацював не випадковим чином, а що він дійсно надає штучній мережі певне уявлення про тренувальний набір даних після попереднього тренування.

4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту

Ідея проекту полягає в створенні компанії з дослідження та розробки технологій штучного інтелекту. Розглянемо зміст ідеї, можливі напрямки застосування, основні переваги, які зможе отримати користувач представлено у таблиці 4.1.

Таблиця 4.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	1. Технологічний консалтинг	оптимізація діяльності технічних підрозділів компаній на основі наявного досвіду, технологічних напрацювань
	2. Аутсорсинг	делегування вирішення питань, пов'язаних з розробкою, впровадженням і супроводом інформаційних систем як цілком на рівні інфраструктури підприємства так і обсягів робіт, пов'язаних з розвитком та підтримкою функціонування окремих ділянок системи.
	3. Розробка та патентування технологій	використання готових передових рішень

Даний проект відрізняється тим, що спеціалізується на певній предметній області.

На ринку наявні конкуренти, які надають схожий набір послуг, але вони не спеціалізуються на конкретній предметній області, що не дозволяє витрачати велику кількість ресурсів на дослідження технологій штучного інтелекту, а отже якість цих послуг може бути нижчою.

Таблиця 4.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-Економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент 1	Конкурент 2	Конкурент 3			
1.	Технічна	-швидкість виконання задач -якісне надання послуг -сучасні методи	-швидкість виконання задач -широкий спектр задач	-широкий спектр задач	-широкий спектр задач -якісне надання послуг	спектр задач вужчий, отже менший ринок		спеціалізація дозволяє надавати послуги на кращому рівні
2.	Економічна	Невеликі витрати на постійне навчання фахівців	Витрати на аналіз технологій	Витрати на аналіз технологій	Витрати на купівлю готових технологій			Невеликі затрати на навчання фахівців
3.	Надійність	Висока якість послуг	Висока якість послуг	Гарантійне обслуговування	Висока якість послуг			якість послуг дозволяє розробляти надійні рішення

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 4.3):

Таблиця 4.3 - Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
	Алгоритм перенесення знання спайкових мереж в штучні мережі	Вже розроблено	Технологія доступна
		Середовище розробки PyCharm, мова Python	Технологія доступна
		Середовище розробки IntelliJ IDEA, мова Java	Технологія доступна
		Рішення від DARPA SyNAPSE	Технологія доступна
		Рішення від IBM TrueNorth	Технологія доступна
Обрана технологія реалізації ідеї проекту: в якості середовища розробки було обрано PyCharm та мову Python зважаючи на швидкість написання та універсальність мови; для апаратної реалізації спайкових мереж обрано рішення від IBM, чіп TrueNorth.			

За результатами аналізу таблиці робиться висновок щодо можливості технологічної реалізації проекту: так чи ні, а також технологічного шляху, яким це доцільно зробити (з поміж названих технологій обираються такі, що доступні авторам проекту та є наявними на ринку).

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводиться аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 4.4).

Таблиця 4.4 - Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од	20000 за рік
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Сертифікація
6	Середня норма рентабельності в галузі (або по ринку), %	60

Ринок є привабливий для входу та потребує новітніх алгоритмів пошуку найкращих, оптимальних методів вирішення задач.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиця 4.5).

Таблиця 4.5 - Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Розробка технологій штучного інтелекту для вирішення низки задач	Потенційна група клієнтів є технологічні компанії	-потреба в продуті оптимізації -рішення які надають конкурентну перевагу підприємству	-якісне надання послуг -технічна консультація та документація

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 4.6 та 4.7). Фактори в таблиці подаються в порядку зменшення значущості.

Таблиця 4.6 - Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	З’являються конкуренти	На ринку утворюються конкуренти з якіснішими послугами	-купівля компанії-конкурента -удосконалення якості продукту
2	Зменшиться попит	Зменшиться попит на новітні розробки штучного інтелекту, будуть використовуватись готові рішення	-маркетинг -виготовлення універсальних програмо-апаратних рішень

Таблиця 4.7 - Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Збільшиться попит	Збільшиться попит на новітні розробки штучного інтелекту	-підвищення ціни -виготовлення універсальних програмо-апаратних рішень -розширення ринку
2	Ріст іноземних інвестицій	Ріст інвестицій призведе до розширення	-розширення ринку

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку (таблиця 4.8).

Таблиця 4.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1.Вказати тип конкуренції - олігополія	Домінує мала кількість фірм	Пришвидшення розроблення унікального продукту
2. За рівнем конкурентної боротьби національний	Боротьба ведеться на національному ринку	Вихід на міжнародний ринок
3. За галузевою ознакою - внутрішньогалузева	Боротьба між товариствами галузі штучного інтелекту	Збільшення якості надання послуг
4. Конкуренція за видами товарів: - товарно-видова	Різновиди однієї категорії товару, які здатні задовольнити конкретне бажання покупця	Відсутня
5. За характером конкурентних переваг - нецінова	Ключовим фактором конкурентної спроможності є експертиза в предметній області	Збільшення кількості наукових розробок
6. За інтенсивністю - не марочна	Не прив’язаний до певної марки, може використовувати будь-де	Співпраця з різними марками

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таблиця 4.9).

Таблиця 4.9-Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Наведені в таблиці 4.2	Можуть з’явитись надаючи більш ширший спектр послуг	IBM	Клієнтам потрібні як готові рішення так і специфічні	Частково присутні
Висновки:	Конкуренти не мають достатньої експертизи	Потенційні конкуренти можуть виконувати простіші замовлення	Для отримання нейроморфних комп’ютерів TrueNorth	Можна виграти на вирішенні специфічних задач	Простіші рішення присутні на ринку

Робота на ринку можлива, попри наявність конкурентів, через досягнуту експертизу в області нейронних мереж конкуренти не здатні на виконання на стільки складних розробок; але простіші завдання конкуренти в змозі виконувати.

На основі аналізу конкуренції, проведеного в таблиця 4.9, а також із урахуванням характеристик ідеї проекту (таблиця 4.2), вимог споживачів до товару (таблиці 4.5) та факторів маркетингового середовища (таблиці 4.6 та 4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за таблицею 4.10.

Таблиця 4.10 - Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Експертиза в предметній області	Експертиза в області дозволяє розв’язувати складні проблеми
2	Можливість розробки рішень для специфічних проблем	Компанія надає не лише послуги розробки, але й досліджень, тому вирішуються проблеми, для яких немає готових рішень на ринку
3	Спектр послуг	Спекрт послуг, які можна отримати при розробці

За визначеними факторами конкурентоспроможності (табл. 4.10) проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 4.11). (С.П. – стартап проект, К.1 – Конкурент 1, К.2 – Конкурент 2)

Таблиця 4.11 - Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспромо жності	Бали (1-20)	Рейтинг товарів-конкурентів у порівнянні з … (Конкурент 1,2,3)						
			-3	-2	-1	0	+1	+2	+3
1	Експертиза в предметній області	20			К.2			К.1	С.П.
2	Можливість розробки рішень для специфічних проблем	20					К.1 К.2		С.П.
3	Спектр послуг	18		К.2				С.П.	К.1

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 4.10).

Таблиця 4.12 - SWOT- аналіз стартап-проекту

Сильні сторони: експертиза в предметній області	Слабкі сторони: вузький спектр послуг
Можливості: збільшення попиту	Загрози: зменшення попиту, конкуренція

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (таблиця 4.9, аналіз потенційних конкурентів).

Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (таблиця. 4.13).

Таблиця 4.13 - Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Отримання основних клієнтів	середня	До 6 місяців
2	Отримання клієнтів з шаблонними задачами	висока	Необмежено
3	Розробка та продаж продуктів штучного інтелекту для вирішення тривіальних задач	висока	До одного року

Після аналізу зазначити обрану альтернативу.

Першою альтернативою є розробка та продаж продуктів штучного інтелекту для вирішення тривіальних задач, на який існує масовий попит.

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (таблиця 4.14).

Таблиця 4.14 -Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Компанії, яким у яких немає експертизи в області штучного інтелекту та потребують певних програмних і/або апаратних рішень	Споживачі готові сприйняти продукт.	Попит є в цільовому сегменті	Конкуренція невелика	Вхід в сегмент не складатиме значних зусиль
Які цільові групи обрано: технологічні компанії				

За результатами аналізу потенційних груп споживачів (сегментів) обираються цільові групи, для яких пропонується товар, та визначається стратегія охоплення ринку:

- якщо компанія зосереджується на одному сегменті – вона обирає стратегію концентрованого маркетингу;
- якщо працює із кількома сегментами, розробляючи для них окремо програми ринкового впливу – вона використовує стратегію диференційованого маркетингу;
- якщо компанія працює із всім ринком, пропонуючи стандартизовану програму (включно із характеристиками товару/послуги) – вона використовує масовий маркетинг.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 4.15).

Таблиця 4.15 -Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Стратегія виклику лідера	Стратегія спеціалізації	Індивідуальний підхід до клієнта; краща якість відповідно до місця реалізації	Стратегія спеціалізації передбачає концентрацію на потребах одного цільового сегменту, без прагнення охопити увесь ринок. Мета тут полягає в задоволенні потреб вибраного цільового сегменту краще, ніж конкуренти.

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 8.16).

Таблиця 4.16 -Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Частково	Шукати нових та забирати споживачів у конкурентів	Ні	Стратегія заняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. табл. 4.14), а також в залежності від обраної базової стратегії розвитку (табл. 4.15) та стратегії конкурентної поведінки (табл. 4.16) розробляється стратегія позиціонування (табл. 4.17). що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 -Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Вирішення нетипових задач штучного інтелекту	Стратегія спеціалізації	Складні задачі Дослідження нових методів Індивідуальні рішення	глибока експертиза, складні задачі, сучасні розробки

Результатом виконання підрозділу є узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап-компанії на ринку.

4.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 - Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	вирішення простих задач за допомогою методів штучного інтелекту	шаблонні та коробочні рішення	добра якість за доступну ціну
2	вирішення складних задач за допомогою методів штучного інтелекту	комплексний та глибокий підхід до вирішення проблеми	ефективне рішення складних задач

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 4.19)

Таблиця 4.19 - Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Надання сервісів в сфері штучного інтелекту		
	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	-		
	Якість: стандарти, нормативи, параметри тестування тощо		
	Технологічний консалтинг, аутсорсинг та розроблені рішення в сфері штучного інтелекту		
	Марка: назва компанії		
	Готовий продукт		
	Технічна підтримка		
Патенти			

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 4.20). Аналіз проводиться експертним методом.

Таблиця 4.20 -Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
50 тис. грн	75 тис. грн	500 тис. грн	10 тис. грн – 100 тис. грн

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 4.21):

- проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);

- вибір та обґрунтування оптимальної глибини каналу збуту;

- вибір та обґрунтування виду посередників.

Таблиця 4.21 -Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Знаходження на сайті послуг, вибір конкретних послуг, обговорення завдання, оплата	-	Виробник-споживач	Web-сайт

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.22).

Таблиця 4.22- Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Знають, які саме послуги треба для вирішення задач	Веб-сайт, телефон, зустрічі	Підтримка, індивідуальний підхід	Донесення переваг до клієнтів	Вирішення задач штучного інтелекту високої складності

Результатом пункту 5 є ринкова (маркетингова) програма, що включає в себе концепції послуг, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні

переваги ідеї, стан та динаміку ринкового середовища, в межах якого буде впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

Висновки: Є можливість ринкової комерціалізації проекту. Попит на послуги присутні, попри наявність продуктів-аналогів. Розроблений проект має свої переваги над конкурентами у вигляді співвідношення ціна - якість. З огляду на потенційні групи клієнтів є перспективи для входження на ринок. Отже конкурентоспроможність послуг висока. В якості базової стратегії розвитку обрана стратегія специфікації. В якості альтернативи впровадження доцільно обрати комбінацію з іншими методами аналізу зображення та розміщення реклами послуг.

ВИСНОВКИ

Під час написання магістерської дисертації було досліджено вплив застосування спайкової нейронної мережі на етапі попереднього тренування при навчанні класичної штучної нейронної мережі. Для цього було проаналізовано компоненти штучних та спайкових нейронних мереж, проаналізовано методи попереднього тренування, розроблено метод попереднього тренування за допомогою спайкових нейронних мереж та визначено компоненти розробленого методу, які дали найвищу точність штучної нейронної мережі.

В якості набору даних було обрано MNIST з метою зробити задачу класифікації простою, для уникнення побудови складної архітектури нейронної мережі. У даній роботі для навчання використовувались лише 600 позначених прикладів з тренувального набору, решта навчального набору (60 000) використовувалась як нерозмічена. Для вимірювання точності було використано 10 000 прикладів тестового набору. Було порівняно різні стратегії перенесення ваг зі спайкової нейронної мережі на традиційну штучну нейронну мережу з використанням різних активаційних функцій. Найкращі з них - це середня нормалізація та стандартизація з використанням функції активації ReLU, які дали точність 91.08% та 91.02%. Без використання запропонованого методу штучна мережа досягла точності 88,66%.

При проведенні експериментів була зроблена перевірка, яка дозволила впевнитись, що даний метод спрацював не випадковим чином, а що він надає штучній мережі певне уявлення про тренувальний набір даних після попереднього тренування.

Попереднє навчання за допомогою спайкових нейронних мереж в таких простих нейронних мережах не вирішує практичних проблеми. Однак запропонований метод може бути розширений для використання в нейронних мережах з більш складними архітектурами (згорткові нейронні мережі, рекурентні нейронні мережі, генеративні змагальні мережі тощо), які використовуються в сучасних системах. Серед недоліків запропонованого підходу можна виділити потребу в налаштуванні параметрів SNN та навчання цієї мережі, що потребує додаткових ресурсів.

Подальші дослідження зосереджені на розробці ефективнішого алгоритму перенесення моделі спайкової нейронної мережі та на застосування запропонованого методу попереднього навчання для більш складних архітектур штучних нейронних мереж. Більш ефективне перенесення моделі повинно зробити початкову точність штучної нейронної мережі такою ж, як і точність щойно навченої спайкової мережі. Перенесення моделі не обмежується лише перенесенням ваг, але вимагає змін у моделі нейронів, алгоритмі навчання та структурі мережі.

Якість спайкової нейронної мережі теж відіграє важливу роль. Існує потреба у більш глибокому вивченні цього типу нейронних мереж. В даний час вони недооцінені і дуже мало досліджень проводяться у цій галузі. Налаштування параметрів спайкової нейронної мережі все ще нелегке і тренування мережі складне; симуляція різних процесів займає багато часу, деякі з яких можуть бути непотрібними для вирішення задач класифікації, а отже, можуть бути спрощені.

Запропонований метод вимагає математичних обґрунтувань, які дозволять визначити границі такого підходу та сприяти побудові більш ефективних алгоритмів перенесення моделі.

Розроблено стартап проект «Дослідження та розробка технологій штучного інтелекту».

ПЕРЕЛІК ПОСИЛАНЬ

1. Ayodele T. Types of Machine Learning Algorithms / Taiwo Ayodele // New Advances in Machine Learning / Taiwo Ayodele., 2010.

2. Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning [Електронний ресурс] – Режим доступу до ресурсу: http://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/.

3. Kamath U. Mastering Java Machine Learning / U. Kamath, K. Choppella., 2017.

4. Goodfellow I. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville., 2016. – (MIT Press).

5. Belkin M. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation / M. Belkin, P. Niyogi // Neural Computation / M. Belkin, P. Niyogi., 2003. – (MIT Press). – С. 1373–1396.

6. Chapelle O. Cluster kernels for semi-supervised learning / O. Chapelle, J. Weston, B. Schölkopf // Advances in Neural Information Processing Systems / O. Chapelle, J. Weston, B. Schölkopf., 2002.

7. Lasserre J. Principled Hybrids of Generative and Discriminative Models / J. Lasserre, C. Bishop, T. Bishop // Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition / J. Lasserre, C. Bishop, T. Bishop., 2006. – (IEEE Computer Society).

8. Larochelle H. Classification Using Discriminative Restricted Boltzmann Machines / H. Larochelle, Y. Bengio // Proceedings of the 25th International Conference on Machine Learning / H. Larochelle, Y. Bengio., 2008. – С. 536–543.

9. Krenker A. Introduction to the Artificial Neural Networks / A. Krenker, J. Bester, A. Kos // Artificial Neural Networks - Methodological Advances and Biomedical Applications / A. Krenker, J. Bester, A. Kos., 2011.

10. Cybenko G. Approximation by superpositions of a sigmoidal function / George Cybenko // Mathematics of Control, Signals and Systems / George Cybenko., 1989.

11. Snyman J. Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms / Jan Snyman., 2005. – (Springer US).

12. Looking inside neural nets [Електронний ресурс] – Режим доступу до ресурсу: https://ml4a.github.io/ml4a/looking_inside_neural_nets/.

13. Neural Network Zoo Prequel: Cells and Layers [Електронний ресурс] – Режим доступу до ресурсу: http://www.asimovinstitute.org/neural-network-zoo-prequel-cells-layers/.

14. Neural Network Zoo [Електронний ресурс] – Режим доступу до ресурсу: http://www.asimovinstitute.org/neural-network-zoo/.

15. Quesada A. 5 algorithms to train a neural network [Електронний ресурс] / Alberto Quesada – Режим доступу до ресурсу: https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network.

16. Kukacka J. Regularization for Deep Learning: A Taxonomy / J. Kukacka, V. Golkov, D. Cremers. – 2017.

17. Bishop C. Neural Networks for Pattern Recognition / Christopher Bishop., 1995.

18. Maxout networks / [I. Goodfellow, D. Warde-Farley, M. Mirza та ін.] // Proceedings of the 30th International Conference on International Conference on Machine Learning / [I. Goodfellow, D. Warde-Farley, M. Mirza та ін.]., 2013.

19. Soni D. Spiking Neural Networks, the Next Generation of Machine Learning [Електронний ресурс] / Devin Soni – Режим доступу до ресурсу: https://towardsdatascience.com/spiking-neural-networks-the-next-generation-of-machine-learning-84e167f4eb2b.

20. Gruning A. Spiking Neural Networks: Principles and Challenges / A. Gruning, S. Bohte // ESANN / A. Gruning, S. Bohte., 2014.

21. Basegmez E. The Next Generation Neural Networks: Deep Learning and Spiking Neural Networks / Erdem Basegmez. – 2014.

22. Paugam-Moisy H. Computing with Spiking Neuron Networks / H. Paugam-Moisy, S. Bohte // Handbook of Natural Computing / H. Paugam-Moisy, S. Bohte., 2012.

23. Lateral inhibition [Електронний ресурс] – Режим доступу до ресурсу: 23. https://en.wikipedia.org/wiki/Lateral_inhibition.

24. Bekolay T. Learning in large-scale spiking neural networks / Bekolay Trevor, 2011.

25. Bohte S. Error-backpropagation in temporally encoded networks of spiking neurons / S. Bohte, J. Kok, H. La Poutre // Neurocomputing / S. Bohte, J. Kok, H. La Poutre., 2002. – С. 17–37.

26. Transfer Learning [Електронний ресурс] – Режим доступу до ресурсу: http://cs231n.github.io/transfer-learning/.

27. Unsupervised Pretraining [Електронний ресурс] – Режим доступу до ресурсу: https://martin-thoma.com/unsupervised-pretraining/.

28. Dorogyy Y. Designing spiking neural networks / Y. Dorogyy, V. Kolisnichenko. // 13th International Conference on Modern Problems of Radio Engineering. Telecommunications and Computer Science (TCSET). – 2016.

29. Bengio Y. Practical recommendations for gradient-based training of deep architectures / Yoshua Bengio. – 2012.

30. Diehl P. “Unsupervised learning of digit recognition using spike-timing-dependent plasticity / P. Diehl, M. Cook // Frontiers in Computational Neuroscience / P. Diehl, M. Cook., 2015.

Додаток А – Тези доповіді «Аналіз моделей спайкових нейронів» на Міжнародні науково-технічі конференції «Сучасні інформаційно-телекомунікаційні технології»

Дорогий Я.Ю.,
Колісніченко В.Ю.,
Національний технічний університет України «Київський Політехнічний Інститут»
м. Київ, Україна

АНАЛІЗ МОДЕЛЕЙ СПАЙКОВИХ НЕЙРОНІВ

Спайкові нейронні мережі – це покоління мереж, які підвищують рівень наближення до біологічних нейронних мереж. Цей вид мереж може бути використаним для оброки інформації, так само як і в традиційних штучних нейронних мережах. Крім того, їх можна використати в неврології, наприклад, для моделювання центральної нервової системи. Спайкова модель нейрона відіграє важливу роль при побудові мережі. У науковій праці досліджуються низка моделей та їх застосування.

Наступним кроком у розвитку штучних нейронних мереж є спайкові нейронні мережі. На відміну від попередніх, для обміну повідомленнями, нейрони в даних мережах використовують імпульси, або спайки – короткочасну зміну напруги. Саме так працюють біологічні нейрони. Одним із перших завдань при проектуванні спайкових мереж є вибір моделі нейрона. Модель нейрона передбачає закони зміни напруги. Найбільш поширеними моделями спайкових нейронів на сьогодні є: Hodgkin-Huxley (HH), Integrate-and-Fire (IF), Theta model, Izhikevich model та Spike Response Model (SRM) [1].

Модель HH являє собою комплекс нелінійний диференційних рівнянь, які характеризують електричний сигнал. Ця модель дозволяє детально моделювати поведінку біологічних нейронів, таку як зміну потенціалу при спрацюванні, період рефакторінгу та відносний період рефакторінгу. Але модель Годжина-Гакслі є складною, тому важко реалізовувати великі мережі використовуючи цю модель.

Модель Integrate-and-Fire простіша ніж HH, а отже, потребує менше ресурсів при обчисленні. Недоліком є те, що вона є менш точною, та не зовсім повторює дію біологічних нейронів. Наприклад, якщо модель отримала недостатній заряд для спрацювання, то вона зберігає цей заряд до наступного разу. Покращена модель Leaky-Integrate-and-Fire усуває цей недолік.

Spike Response Model є узагальненням моделі LIF. Ця модель дозволяє симулювати складні обчислення, але зі значно простішою структурою, ніж в HH. При реалізації мережі, не завжди потрібне повне повторення характеру зміни потенціалу мембрани нейрона, тому можна спростити SRM та отримати значно швидшу модель [2]. Мембранний потенціал нейрона спрощеної моделі SRM представлений на рисунку 1.

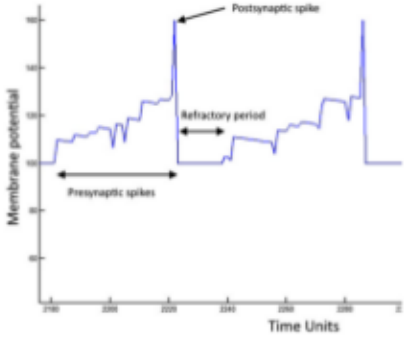

Рисунок 1 – Імпульс нейрона спрощеної моделі SRM

Дана модель отримала перевагу у швидкості над моделлю SRM для класифікації зображень, при однаковій похибці. Наша ідея полягає в тому, щоб дослідити варіанти

спрощення функціоналу зміни потенціалу мембрани нейрона в залежності від вхідних даних та типу задачі, що розв'язується.

Література

1. The Next Generation Neural Networks: Deep Learning and Spiking Neural Networks / Erdem Basegmez. – 2014. – p. 37.
2. Iakymchuk T. Simplified spiking neural network architecture and STDP learning algorithm applied to image classification / T.Iakymchuk, A.Rosado-Muñoz, J.F.Guerrero-Martínez, M.Bataller-Mompeán, J.V.Francés-Villora // EURASIP Journal on Image and Video Processing 2015, 2015:4. – 2015.

Додаток Б – Тези доповіді «Дослідження методів тренування спайкових нейронних мереж» – на II Міжнародній науково-технічній конференції «Актуальні проблеми розвитку науки і техніки»

Дорогий Я.Ю.,
Колісніченко В.Ю.,
Національний технічний університет України «Київський Політехнічний Інститут»
м. Київ, Україна

ДОСЛІДЖЕННЯ МЕТОДІВ ТРЕНУВАННЯ СПАЙКОВИХ НЕЙРОННИХ МЕРЕЖ

Спайкові нейронні мережі відрізняються від звичайних нейронних мереж, тому не можна використовувати класичні методи для їх тренування (наприклад backpropagation). Тематика спайкових мереж досить нова, тому не існує консенсусу як найкраще навчати мережі. У доповіді розглядаються методи тренування спайкових нейронних мереж для вирішення задач машинного навчання.

Різноманіття методів навчання спайкових мереж можна розділити на два типи: методи, побудовані на основі зворотного поширення помилки та методи, засновані на біологічних законах (правилах).

Традиційні алгоритми для навчання штучних мереж можна адаптувати для навчання спайкових мереж, використовуючи часове кодування (temporal coding). Одним з таких методів є SpikeProp. Він використовує SRM модель нейрона для передачі інформації під час імпульсів [1, с 35]. Також існують алгоритми, які використовують моделі нейронів QIF та Theta [2, с 26]. Методи FreqProp та Remote Supervised Method (ReSuMe) беруть ідеї з біологічних нейронних мереж [3, с 46].

Навчання спайкових мереж можна робити за допомогою законів, відкритих в області нейронаук. Найчастіше імітують Spike-timing-dependent plasticity (узагальнення правила Хебба) - процес встановлення ваг (сил зв'язку) між нейронами. Ще один алгоритм використовують для навчання мереж - Local Hebbian delay-learning [1, с 33].

При використання різних методів навчання потрібно зважати на те, які задачі ставлються. Потрібно зважати на швидкість навчання спайкової нейронної мережі та на втрату точності (порівняно зі звичайними ANN) для задач розпізнавання та класифікації. Одна із цілей поставлена на спайкові мережі — це імітування роботи біологічних нейронних мереж, тому й методи їх навчання повинні бути схожі.

Література:

1. Developing a supervised training algorithm for limited precision feed-forward spiking neural networks / Evangelos Stromatias. – 2011.
2. The Next Generation Neural Networks: Deep Learning and Spiking Neural Networks / Erdem Basegmez. – 2014.
3. Learning in large-scale spiking neural networks / Trevor Bekolay. - 2011.

Додаток В – Стаття «Designing Spiking Neural Networks» в «Proceedings of the XIIIth International Conference TCSET'2016»

Designing Spiking Neural Networks

Yaroslav Dorogyy, Vadym Kolisnichenko

Abstract – **The problem of design is the most important part of complex systems building. This is also true for spiking neural networks. In this paper, the next steps of SNN design are described: coding, selecting neuron model and learning algorithm, creating network architecture. Software and hardware solutions for simulating these networks are also discussed. We propose a range of evolution directions, future studies on every step of the design. Our methods are based on detailed analysis of existing solutions and needs.**

Keywords – **Spkiking Neural Network, Spike Coding, STDP, Neurons Modeling, Biological Neuron Models.**

I. Introduction

Spiking Neural Networks (SNN) are the third generation neural network models. This kind of network is more realistic then usual artificial neural network, because it simulates internal neuron behavior by firing impulses (spikes). Choosing the right network parameters is the main task of the network design, which consists of several steps. Practical modeling experiments or network deployment could be done with different software and hardware solutions.

II. Coding

Data coding is important part of the network design, because SNN is impulse-based network, so information could not be represented in a usual manner. The data should be transformed in the series of spikes.

The neuronal coding methods can be divided into three categories [1]: Rate Coding, Temporal Coding, Population Coding.

Rate coding is a traditional coding scheme, assuming that most, if not all, information about the stimulus is contained in the firing rate of the neuron. Because the sequence of action potentials generated by a given stimulus varies from trial to trial, neuronal responses are typically treated statistically or probabilistically.

When precise spike timing or high-frequency firing-rate fluctuations are found to carry information, the neural code is often identified as a temporal code. A number of studies have found that the temporal resolution of the neural code is on a millisecond time scale, indicating that precise spike timing is a significant element in neural coding.

In population coding, each neuron has a distribution of responses over some set of inputs, and the responses of many neurons may be combined to determine some value about the inputs.

Yaroslav Dorogyy, Vadym Kolisnichenko - The National Technical University of Ukraine "Kyiv Polytechnic Institute", Prospect Peremogy Str, 37, Kiev, 03056, UKRAINE, E-mail: alt2600h@gmail.com

III. Neuron Model

A neuron model provides a law of changing its membrane potential. The most common models today are Hodgkin-Huxley (HH), Integrate-and-Fire (IF), Theta model, Izhikevich model and SRM.

HH Model is a set of nonlinear differential equations that characterize the electrical signal. This model allows simulating the detailed behavior of biological neurons, such as the change in potential drawdown period, refactoring and relative refactoring periods. However, the HH model is complex, so it is difficult to implement large networks using this model.

Integrate-and-Fire type is simpler than HH, and therefore requires fewer resources in the calculation. The downside is that it is less accurate and does not repeat the exactly action of biological neurons. For example, if the neuron got insufficient charge, it saves the potential until next time. Improved model Leaky-Integrate-and-Fire eliminates this drawback.

Spike Response Model is a generalization of the LIF model. This model allows us to simulate complex calculations, but with a much simpler structure than HH. An example of SRM is presented in Eq. (1).

$$u(t) = \eta(t-\tau) + \int_{-\infty}^{+\infty} k(t-\tau, s)(t-s)ds \qquad (1)$$

It is not always important to repeat the full nature of the neuron membrane potential changes, so it is possible to simplify SRM and get faster model [2]. An example of membrane potential changing of such model is presented in Fig. 1.

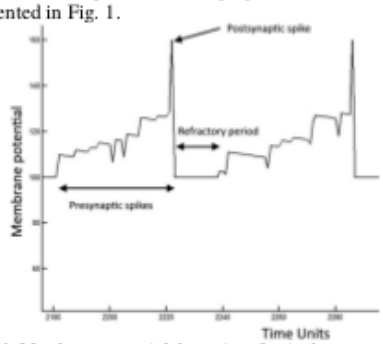

Fig.1. Membrane potential dynamics of a single neuron with simplified membrane model

This model has the advantage of speed over the SRM model for image classification with the same level of accuracy.

Izhikevich model can exhibit firing patterns of all known types of cortical neurons with the choice of parameters. It takes only 13 floating point operations to

simulate 1 ms of the model, so it is quite efficient in large-scale simulations of cortical networks.

Some learning methods are made purely for certain kinds of neuron models. If such method is chosen, the step of selecting neuron model could be passed. Some of such learning algorithms are described in the next section.

Choosing the most efficient model requires a detailed analysis of the problem and analysis of advantages and disadvantages of different models.

IV. Learning Algorithm

Spiking neural networks differ from usual neural networks, so classical methods could not be used for their training (eg backpropagation). This subject is new, so there is no consensus on how to train SNN. There are two ways of training SNN: 1) adapt traditional learning methods to SNN; 2) develop new learning algorithms purely for this kind of networks [3].

Different algorithms could be used: SpikeProp, FreqProp, Remote Supervised Method (ReSuMe), Local Hebbian delay-learning and etc.

SpikeProp is supervised learning algorithm, which is derived from traditional error-back-propagation. It can perform complex non-linear classification in fast temporal coding just as well as rate-coded networks [4].

Remote Supervised Method is efficient learning algorithm that integrates integrates the idea of learning windows (from spike-based Hebbian rules) with a concept of remote supervision. The experiment presented in [5] verified that ReSuMe can efficiently learn the desired temporal sequences of spikes and that the learning process converges quickly. This method works well with different neuron models, that practically confirmed in [6].

Frequency-based error back-propagation is biologically plausible method that uses Hebbian rules to establish synaptic weights. Error back-propagation uses a level of confidence, which is based on familiarity detection neurons.

If the task is to simulate biological neural network, it is important to use rules and laws discovered in neuroscience (eg Spike-Timing-Dependent Plasticity, Hebbian rules), so such network will behave exactly as biological.

Spike-timing-dependent plasticity is a process of establishing weights of connections between neurons. The rule is simple: if postsynaptic firing occurs after presynaptic firing, the weight becomes stronger; if postsynaptic firing occurs before presynaptic firing, the weight is decreased. The weight change is described in Eq. (2).

$$\Delta w_j = \sum_{t=1}^{N}\sum_{n=1}^{N} W(t_i^n - t_j^f) \qquad (2)$$

The weight change of a synapse from a presynaptic neuron j depends on the relative timing between presynaptic spike arrivals and postsynaptic spikes. Presynaptic spike arrival times at synapse j by t_j^f where f=1,2,3,... counts the presynaptic spikes. Similarly, t_n with n=1,2,3,... labels the firing times of the postsynaptic neuron, W(x) denotes one of the STDP functions (also called learning window). Usually the W(x) is chosen as in Eq. (3) (for positive x).

$$W(x) = A_+ \exp(-x/\tau_+) \qquad (3)$$

The parameter A_+ may depend on the current value of the synaptic weight w_j. The time constant is approximately τ_+=10ms.

Many approaches for configuring spiking neural networks rely on mapping pre-trained artificial neural networks onto spiking neural networks using a firing rate code [7]. Such methods has advantages and disadvantages. It is possible to use existing machine learning frameworks to train an artificial networks and map them on spiking neural networks. It is very easy to implement and the accuracy is relatively high, but it is not suitable for simulating real-time learning. The best use of these methods is for neuromorphic platforms.

It is also important to mention that nowadays classification accuracy in SNN is lower than in ANN, but they are candidates to be faster than traditional ANN. Depending on the problem is, various methods could be selected, as in one case, to simulate biological neural network, and another to reduce speed of training network for image recognition.

V. Network Architecture

The hardest and the longest step is creating network architecture. It includes not only theoretical aspects, but also practical experiments.

This step requires determining the number of layers, number of neurons in each layers. As input data complexity is increased, the number of layers should be also increased to overcome undefitting.

Spiking neural networks could be also combined with constitutional neural networks or deep belief networks or other. Choosing such networks requires selecting more parameters. Some methodologies of making CNN better are already described [8].Combining with other networks could increase accuracy of the SNN. It can also increase the level of similarity with biological neural network.

The "spiking" part of SNN should be also tuned. Depending on the model of neuron, different constants (parameters) should be set. This part requires many experiments to choose best values. Different methodologies are described in [9].

VI. Software and Hardware Spiking Network Modeling

There is a wide range of software simulators; they could be divided into 3 groups. The first type of software usually supports the simulation of complex neural models with a high-level detail and accuracy. However, modeling networks is very time-consuming.

The next software belongs to this group: GENESIS, NEURON, NEST and Brian.

Software that can solve machine learning tasks belongs to the second group. The most popular is SpikeNet. SpikeNET is a program designed for simulating very large networks of asynchronous spiking neurons. Neurons are simulated with a limited number of parameters that includes classic properties like the post-synaptic potential and threshold, but also more novel features like dendritic sensitivity. SpikeNET can be used to simulate networks with millions of neurons and hundreds of millions of synaptic weights. Optimization of computation time and the aim of real time computation has been one of the driving forces behind the development of SpikeNET. This software does not allow the simulation of very complex or biologically-realistic neural models.

Software which provides capabilities to support the simulation of relatively complex neural models efficiently so that it can also be convenient for information processing tasks. This software can exploit biological neuron characteristics to perform computation functions and at the same time allows the study of the functionality of these neural characteristics. EDLUT is an instance of such group. The EDLUT (Event-Driven simulator based on Look-Up-Tables) is an advanced tool that allows the simulation of biologically plausible spiking cell models by using two different strategies: time-driven and event-driven based on look-up tables. In this way, EDLUT can highly speed up the simulation process by avoiding the resolution of the differential equations which usually regulate the evolution of the biological system state [10].

The most popular hardware solutions are SpiNNaker (Spiking Neural Network Architecture) and TrueNorth (from IBM). The main aim of SpiNNaker is simulation of human brain. TrueNorth's primary purpose is pattern recognition; while critics say the chip isn't powerful enough, its supporters point out that this is only the first generation, and the capabilities of improved iterations will become clear [11].

It is important to understand that every simulator is limited and some neuron models could not be implemented. Some learning algorithms are also incompatible with existing solution.

VII. Conclusion

In this paper the spiking neural networks design steps are described. The process of design consists of theoretical and practical approaches. Deep analysis of the problem (task) could make spiking neural network more efficient. Depending on the problem different models, algorithms and software must be used.

Future studies of spiking neural network will introduce some changes in design. The coding step will not be changed and SNNs will use existing methods. Whether biological neurons use rate coding or temporal coding is a topic of intense debate within the neuroscience community.

New mathematical approaches and algorithms must be found to make calculations simpler and modeling more accurate. Some neuron models could be simplified, so they will use fewer resources. Simplified models are implemented even today [2]. To use spiking neural networks for real tasks, an efficient learning algorithm must be also discovered.

Development of neuroscience would give us a new methods to build neural networks. However, a lot of rules are discovered, there are many unknown processes in the brain which would help us to better understand a neural network learning algorithm. Network architecture should be also copied from a brain.

To simulate detailed brain activity more computation resources are needed. Moore's law says that the number of transistors in a dense integrated circuit doubles approximately every two years. So, in 10 years we will increase computational resources in 5 times. Parallel computing can also play a role, that the network will be distributed on a set of machines.

A perfect final solution would be a hardware solution, a small chip, which could be plugged into any machine, so it will be able to solve human brain tasks. This chip could be a wireless client that connects to powerful servers, which make calculations.

Nowadays traditional neural networks are widely used, they are accurate and fast enough for current tasks. A transition to spiking neural and evolution of them will happen when traditional neural networks could not solve assigned task.

References

[1] B. Meftah, O. Lezoray and A. Benyettou, "Segmentation and edge detection based on spiking neural network model", *Lecture Notes in Computer Science*, vol. 4682, pp. 26-34, Aug. 2007.

[2] Iakymchuk T. "Simplified spiking neural network architecture and STDP learning algorithm applied to image classification", *EURASIP Journal on Image and Video Processing*, pp. 1-11, Feb. 2015.

[3] Erdem Basegmez, "The Next Generation Neural Networks: Deep Learning and Spiking Neural Networks", *Advanced Seminar in Technical University of Munich*, pp. 1-40, Jun. 2014.

[4] Sander M. Bohte, Joost N. Kok, Han La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons", *Neurocomputing*, vol. 48 pp. 17-37, Jun. 2001.

[5] Filip Ponulak, "ReSuMe – new supervised learning method for Spiking Neural Networks", *Technical report in Institute of Control and Information Engineering*, pp. 1-10, 2005

[6] Andrzej Kasinski and Filip Ponulak, "Experimental Demonstration of Learning Properties of a New Supervised Learning Method for the Spiking Neural

Networks", *Lecture Notes in Computer Science*, vol. 3696, pp 145–152, 2005

[7] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing", *IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2015.

[8] Patrice Y. Simard, Dave Steinkraus, John C. Platt, "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis, *Microsoft Research*, pp. 1-6, Aug. 2003.

[9] Kristofor D. Carlson, Jayram Moorkanikara Nageswaran, Nikil Dutt and Jeffrey L. Krichmar, "An efficient automated parameter tuning framework for spiking neural networks", *Neuromorphic Engineering Systems and Applications*, vol. 8, Feb. 2014.

[10] Ros, E. Carrillo, R, Ortigosa, E. M., Barbour, B, Agís, R. "Event-Driven Simulation Scheme for Spiking Neural Networks Using Lookup Tables to Characterize Neuronal Dynamics", *Neural Computation*, vol. 18, pp. 2959-2993, Dec. 2006

[11] Markoff, John, "A new chip functions like a brain, IBM says", New Yourk Times, p. B1, Aug. 2014.